VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Řízení rozhraní Ethernet pro telemedicínské aplikace

Ethernet interface control for telemedicine applications

2018 David Sojka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **David Sojka**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Řízení rozhraní Ethernet pro telemedicínské aplikace
Ethernet Interface Control for Telemedicine Applications

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Studium standardu datového přenosu po síti Ethernet.
2. Seznámení se s technikou FPGA a možnostmi připojení rozhraní Ethernet k programovatelné logice.
3. Návrh koncepce demonstrační úlohy pro přenos strukturovaných dat po síti Ethernet.
4. Návrh a implementace logického obvodu pro řízení rozhraní Ethernet na Xilinx FPGA.
5. Experimentální testování přenosu vybraných dat v intranetové síti.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] DOSTÁLEK, Libor a Alena KABELOVÁ. *Velký průvodce protokoly TCP/IP a systémem DSN.* Praha: Computer Press, 2000. ISBN 80-7226-323-4.
[2] KAŠÍK, Vladimír. *Programování hradlových polí.* Učební text a návody do cvičení. Ostrava: VŠB-TU Ostrava, 2012.
[3] PINKER, Jiří a Martin POUPA. *Číslicové systémy a jazyk VHDL.* 1. vyd. Praha: BEN - technická literatura, 2006. 349 s. ISBN 80-7300-198-5.
[4] KHALILZAD, Nima et al. FPGA implementation of real-time Ethernet communication using RMII interface. In: *2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN), 2011.* Xi'an, China: IEEE, 2011. DOI: 10.1109/ICCSN.2011.6013943· Print ISBN 978-1-61284-485-5. eISBN 978-1-61284-486-2.
[5] DIGILENT, Inc. *Nexys 4 DDR - Getting Started with Microblaze Servers.* [online] Dostupné z: https://reference.digilentinc.com/learn/programmable-logic/tutorials/nexys-4-ddr-getting-started-with-microblaze-servers/start.
[6] DIGILENT, Inc. *Pmod NIC100: Network Interface Controller* [online] Dostupné z: http://store.digilentinc.com/pmod-nic100-network-interface-controller/.
[7] XILINX, Inc. *AXI Ethernet Lite.* [online] Dostupné z: https://www.xilinx.com/products/intellectual-property/axi_ethernetlite.html#overview.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Vladimír Kašík, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

V Ostravě dne 30. 4. 2018 .....

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Vladimíru Kašíkovi za konzultace, vstřícnost, cenné rady a připomínky během vypracování této práce.

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací logického obvodu pro řízení rozhraní Ethernet na vývojové desce FPGA Digilent Nexys 4 DDR.

V práci je popsána Ethernetová komunikace podle standardu IEEE 802.3 s využitím komunikačních protokolů z rodiny protokolů TCP/IP. Dále je v práci obsažen postup realizace logického obvodu s využitím IP bloků a vývoj aplikačního programu v jazyce C, běžícího na vestavěném procesoru MicroBlaze. Práce také zahrnuje otestování vytvořené demonstrační úlohy.

Klíčová slova

Ethernet, FPGA, TCP/IP, Digilent Nexys 4 DDR, Telemedicína, Biomedicína

Abstract

This bachelor thesis deals with the design and implementation of a logic circuit for Ethernet interface control on the FPGA Digilent Nexys 4 DDR development board.

The thesis describes the IEEE 802.3 Ethernet communication using communication protocols from the TCP/IP protocol family. In addition, the thesis involves the process of realizing the logic circuit based on IP blocks and the development of the application program in C programming language, running on the embedded MicroBlaze soft-core processor. The thesis also includes testing of the demonstration task.

Key words

Ethernet, FPGA, TCP/IP, Digilent Nexys 4 DDR, Telemedicine, Biomedical

Obsah

Seznam použitých zkratek

BSP – Board Support Package

CT – Computer Tomography

CLB – Configurable Logic Block

CRC – Cyclic Redundancy Check

DHCP – Dynamic Host Configuration Protocol

EEG – Elektroencefalografie

FPGA – Field Programable Gate Array

fMRI – functional Magnetic Resonance Imaging

fNIRS – functional Near-InfraRed Spectroscopy

GUI – Graphical User Interface

GPIO – General Port Input/Output

HW – Hardware

IEEE – Institute of Electrical and Electronics Engineers

IP – Internet Protocol

IP – Intellectual Property

IOB – Input/Output Block

LSB – Least Significant Bit

LUT – Look Up Table

lwIP – lightweight Internet Protocol

MAC – Media Access Control

MDIO – Management Data Input/Output

MII – Media Independent Interface

MRI – Magnetic Resonance Imaging

MSB – Most Significant Bit

MTU – Maximum Transmission Unit

NIC – Network Interface Controller

PCB – Protocol Control Block

PET – Positron Emission Tomography

PHY – Physical layer

QoS – Quality of Service

RMII – Reduced Media Independent Interface

RTT – Round Trip Time

RX – Receive

SDK – Software Development Kit

SFD – Start Frame Delimeter

SPECT – Single-Photon Emission Computer Tomography

SPI – Serial Peripheral Interface

SW – Software

TCP/IP – Transmission Control Protocol/Internet Protocol

TCP – Transmission Control Protocol

TTL – Time To Live

TX – Transmit

UDP – User Datagram Protocol

USB – Universal Serial Bus

Seznam obrázků

Seznam tabulek

Úvod

V současné medicíně existuje mnoho přístrojů, zejména diagnostických, které jsou schopny produkovat obrovská množství dat. Tato data je nutné přenášet mezi různými zařízeními, jako jsou například servery, diagnostické přístroje, počítače, vestavěné (embedded) systémy a další, za účelem jejich následné archivace, sdílení, zpracování nebo vizualizace. Samotný přenos informací by měl být co nejrychlejší, aby byla data ze svého úložiště přístupná v podstatě kdykoli a téměř odkudkoli, a aby přenos nezpomaloval další práci s daty.

Pro rychlý přenos dat je vhodná technologie Ethernet, která nabízí různé přenosové rychlosti od jednotek Mbit/s, až po jednotky Gbit/s. Výhodou Ethernetu je také jeho rozšířenost a dostupnost přenosových médií.

Zprostředkování přenosu dat však není jediným úskalím této problematiky. Biomedicínská data musí být také co nejefektivněji zpracována, ideálně v reálném čase. K tomuto účelu se využívají programovatelná hradlová pole FPGA (Field Programmable Gate Array), která zpracovávají digitální signály na úrovni hardwaru, tudíž rychlost zpracování není nijak omezována taktovací frekvencí procesoru.

Cílem této bakalářské práce je navrhnout a implementovat Ethernetové rozhraní na přípravku FPGA Digilent Nexys 4 DDR, za pomoci kterého bude zprostředkován přenos strukturovaných dat mezi již zmíněným přípravkem a počítačem. Dále pak vytvořit demonstrační úlohu, která názorně demonstruje správnost přenesených dat.

V teoretické části této práce je popsán princip Ethernetové komunikace podle standardu IEEE 802.3. Dále je zde uveden popis referenčního modelu ISO/OSI spolu s jednotlivými protokoly z rodiny protokolů TCP/IP, podle kterých je komunikace realizována v praktické části. Následuje stručný popis vývojové desky FPGA Digilent Nexys 4 DDR s možnostmi připojení Ethernetového rozhraní.

Praktická část zahrnuje především návrh logického obvodu za pomoci IP bloků ve vývojovém prostředí Vivado. Součástí logického obvodu je integrovaný procesor MicroBlaze, který představuje řídící jednotku Ethernetové komunikace. Na tomto procesoru je následně postaven aplikační program, ve kterém lze využívat komunikačních protokolů z volně dostupného lwIP stacku. Na konci práce je znázorněna výsledná podoba demonstrační úlohy a otestování doby odezvy vytvořeného serveru.

1 Komunikace po Ethernetové síti a její využití s FPGA v biomedicíně

Tato kapitola zahrnuje současnou problematiku zpracování a přenosu obrovského množství dat z diagnostických přístrojů. Vhodným řešením je využití programovatelných hradlových polí FPGA společně s Ethernetovým rozhraním, které dokáže standardně zprostředkovat rychlost přenosu až 100 Mbit/s. Kapitola obsahuje rozbor Ethernetové komunikace, která se řídí podle normy IEEE 802.3. Aby bylo možné po síti komunikovat, je zapotřebí znát jednotlivé protokoly z rodiny TCP/IP, které jsou níže popsány.

1.1 Současný stav

Lékařská zobrazovací zařízení zaujímají stále důležitější úlohu ve zdravotnictví. Hlavním cílem ve zdravotnictví je včasná a především rychlá diagnostika pacientova onemocnění a také snížení dalších nákladů spojených s touto diagnózou. Pro diagnózu se stále častěji využívají neinvazivní prostředky, které produkují obrovské množství dat, která musí být zpracována a přenášena na jiná pracoviště. Pro splnění těchto cílů se vývojáři lékařských zařízení obrací na programovatelná logická zařízení, jako jsou FPGA (Field Programmable Gate Array) různých firem. V dnešní době existuje široké spektrum technik, které jsou schopny produkovat obrovské množství dat, která musí být následně zpracována, za pomoci moderních algoritmů. Cílem je však zpracovávat biomedicínská data pokud možno v reálném čase, přenášet je rychle na úložiště a mít k nim přístup v podstatě kdykoli a téměř odkudkoli. Tímto tématem se zabývá telemedicína, která poskytuje zdravotnické služby na velké vzdálenosti, prostřednictvím komunikačních a informačních technologií. Jejím cílem je zkvalitnit zdravotní péči, zlepšit přístup ke zdravotnickým službám a podporovat vzdělání a výzkum. Telemedicína se rozděluje na čtyři oblasti, přičemž v této práci se budeme zabývat pouze přenosem informací. [23], [28].

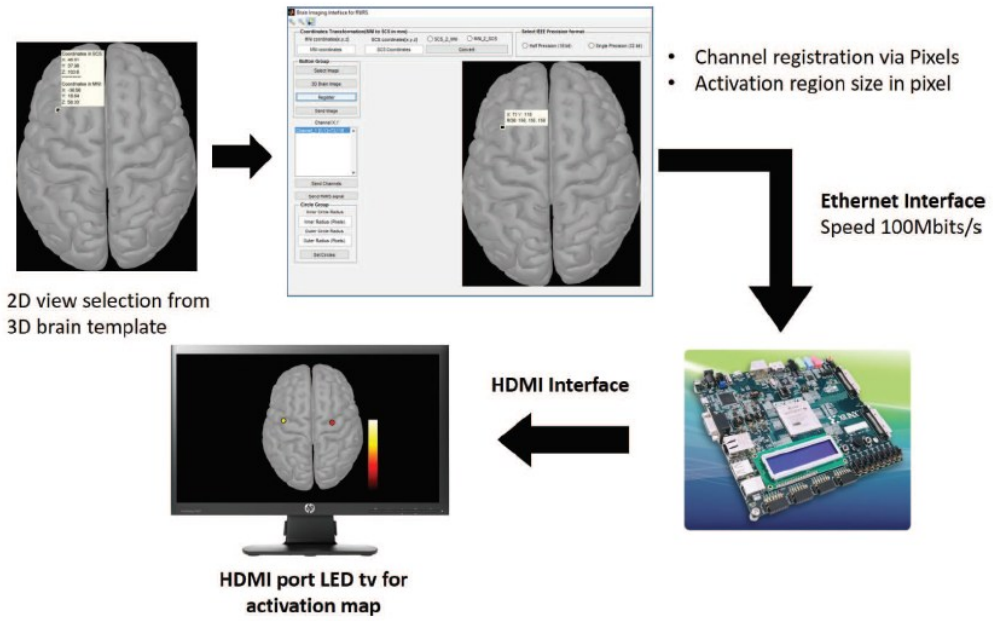
1. Přenos informací,
2. Monitorování na dálku,
3. Dálková terapie,
4. Telemedicínské vzdělávání.

Hlavními producenty obsáhlých množství dat jsou diagnostické přístroje, mezi které se řadí např:

- EEG,
- Ultrazvuk,
- Echo,
- Doppler,
- MRI,
- CT,
- PET,
- SPECT,
- a další.

Autor článku [24] popisuje monitorovací systém, který je založený na FPGA neboli programovatelném hradlovém poli. Bylo zde využito rozhraní HDMI (High Definition Multimedia Interface) k zobrazení anatomie mozku v reálném čase na LED/LCD monitoru. Pro zrychlení zobrazování byla hodnota RGB každého pixelu transformována na jednobytovou informaci a poslána desce FPGA přes rozhraní Ethernet, které bylo navrženo samotnými autory. FPGA ukládá data do paměti, zobrazuje obraz mozku na obrazovce a je připraveno měnit v reálném čase barvy jednotlivých pixelů v závislosti na mozkové

aktivitě. Na obrázku 1 je znázorněn přenos dat pomocí Ethernetu a jejich interpretace prostřednictvím rozhraní HDMI.

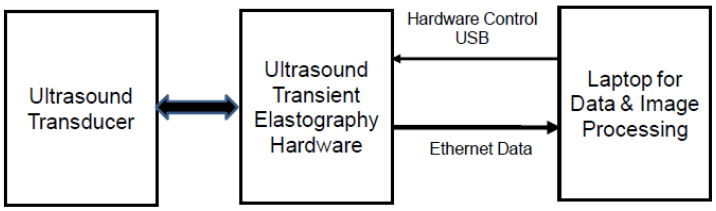


Obrázek 1: 2D zobrazení 3D modelu mozku pro vytvoření mapy mozkové aktivity [24]

Byla zde využita přenosová rychlost Ethernetu 100 Mbits/s s využitím počítače a FPGA Virtex-5 LX50T, obsahující 100 MHz hodiny. Data jsou přijímána přes Ethernetový modul do desky FPGA, ukládána do paměti Block RAM nebo paměti DDR2. Poté co jsou tato obrazová data uložena, prostřednictvím HDMI jsou zobrazována na LCD monitoru. Čas přenosu by mohl být snížený při zvýšení přenosové rychlosti dat Ethernetu na 1 Gbit/s. Tento princip s využitím FPGA, HDMI a Ethernetu může fungovat také pro fMRI (funkční magnetická rezonance), EEG (elektroencefalografie), fNIRS (funkční blízká infračervená spektroskopie) a další vyšetřovací metody, u kterých je důležité sledování obrazu v reálném čase.

V článku [25] autor představuje novou generaci vysoce škálovatelné ultrazvukové platformy pro všechny druhy aplikací s hlavním využitím ve vysokofrekvenčním biomedicínském zobrazování. Tato platforma byla založena na základě desky FPGA Virtex-6 a mikroprocesorového jádra MicroBlaze. Data jsou přenášena z FPGA do PC pomocí Gigabitového Ethernetu, který je součástí desky. Autor uvádí, že tento hardware by mohl být v medicíně použit pro zobrazování ultrazvukových vyšetření pro snímače se střední frekvencí až 20 MHz.

Autor článku [26] popisuje detekci nádorových onemocnění pomocí ultrarychlého skeneru. Neinvazivní lékařská diagnóza je prováděna různými postupy, jako je ultrasonografie, mamografie, CT (počítačová tomografie) vyšetření, MRI (magnetická rezonance) atd. Pro detekci tumoru je vyžadován ultrarychlý skener, který shromažďuje obrovské množství dat pro interpretaci. Vzhledem k tak velkému množství je zapotřebí zabezpečit rychlý přenos pomocí Gigabitového Ethernetového rozhraní. Následující schéma (Obrázek 2) znázorňuje propojení ultrazvukového měniče s FPGA, který jeho činnost řídí. Tento hardware odesílá získaná data do počítače, na zpracování a zobrazování, a jeho funkce je implementována z počítače skrze USB kabel.



Obrázek 2: Blokové schéma ultrarychlého skeneru [26]

Přístroj pracující na principu ultrazvukové elastografie pracuje při frekvenci snímků přesahující 1000 snímků za sekundu. Výpočty potřebné rychlosti přenosu v různých případech jsou uvedeny v tabulce 1.

Tabulka 1: Potřebná přenosová rychlost [26]

Ultrazvuková frekvence	**Vzorkovací frekvence**	**Počet bitů na vzorek**	**Počet kanálů senzoru**	**Přenosová rychlost**
6 MHz	60 M vz/s	12	64	~44 Gb/s
6 MHz	40 M vz/s	8	64	~22 Gb/s
6 MHz	24 M vz/s	8	64	~14 Gb/s
4 MHz	24 M vz/s	8	32	~8 Gb/s

Data z ultrarychlého skeneru jsou přenášena pro zpracování prostřednictvím Ethernetového rozhraní. Signály z kanálů senzorů jsou přijímány pomocí FPGA a přeměněny na pakety. Využívá se zde rychlejšího UDP (User Datagram Protocol) protokolu, IP (Internet Protocol) protokolu a Ethernetového rámce. Celá struktura odesílaného paketu je znázorněna na obrázku 7, s rozdílem, že místo protokolu TCP je v tomto projektu využito protokolu UDP.

Přenášena raw data (surová data) mohou být za pomoci telemedicíny analyzována a zobrazována pomocí GUI (Graphical User Interface) rozhraní notebooku, na různých místech světa. Standardní rozhraní Ethernet umožňuje zpracování dat v přenosném počítači, což snižuje celkové náklady a velikost hardwaru.

1.2 Ethernet

Tato kapitola se zabývá Ethernetovým standardem IEEE 802.3. Popisuje jeho vznik, značení jednotlivých verzí, v závislosti na jeho parametrech, a režimy komunikace, ve kterých je schopen komunikovat s dalšími zařízeními.

1.2.1 Vznik a vývoj Ethernetu

Prvním vynálezcem Ethernetu byl Robert Metcalfe v roce 1973. Původní Ethernet měl rychlost 2,94 Mbit/s. V průběhu následujících deseti let byl vylepšován firmami DEC, Intel a Xerox a poprvé použit pro komerční účely. V roce 1983 byl standardizován normou IEEE 802.3. Tato verze byla označena jako 10BASE5 a podle tohoto označení zvládla přenosovou rychlost 10 Mbit/s skrze hrubý koaxiální kabel. V roce 1987 se začala využívat nestíněná kroucená dvojlinka jako přenosové médium, které bylo spolu s konektorem RJ-45 značně levnější a nahradilo koaxiální kabel. V současnosti existuje více verzí Ethernetu, které se liší způsobem přenosu a hlavně přenosovou rychlostí. V dnešní době se můžeme setkat také s dostupným Gigabitovým Ethernetem, ale existují i rychlejší verze jako například

Terabitový Ethernet (TbE), který označuje typ Ethernetu s vyšší přenosovou rychlostí než 100 Gbit/s [22].

1.2.2 Ethernetové verze

Značení verzí Ethernetu vyplývá z normy IEEE 802.3. V tomto značení se uvádí přenosová rychlost, typ modulace a použité přenosové médium (Obrázek 3).

Obrázek 3: Ethernetové značení (vlastní)

Přenosová rychlost: udává hodnotu rychlosti přenosu dat v Mbit/s.

Přenosové pásmo: udává, jak jsou data modulována na médium. Téměř všechny verze Ethernetu dnes využívají základního přenosového pásma (BASE).

Přenosové médium: označuje typ média, které je pro danou verzi použito, např. TX – kroucená dvojlinka, F – optický kabel, číslo (bez pomlčky) – koaxiální kabel.

1.2.3 Režimy komunikace

Ethernetová komunikace může fungovat ve dvou režimech. Prvním, avšak méně efektivním řešením je poloduplexní režim neboli half duplex. Efektivnější přenos zajišťuje plně duplexní režim (full duplex) a to z důvodu, že je schopen přijímat a vysílat data současně. Podrobnější popisy obou režimů jsou popsány v následujících odstavcích.

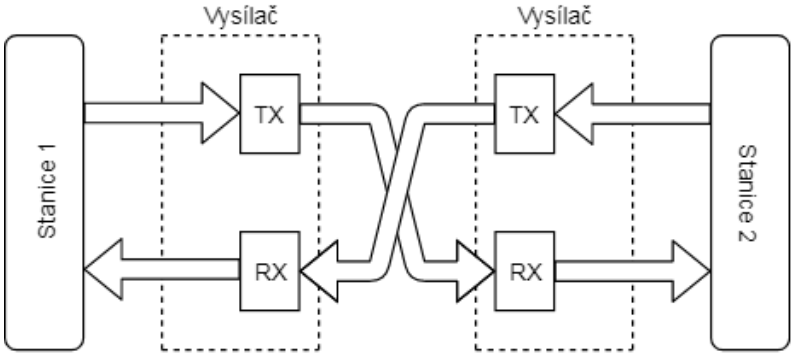
- **Poloduplexní režim**

Poloduplexní režim pracuje se signálem v základním pásmu přes koaxiální kabel nebo skrze hub (rozbočovač), komunikace neumožňuje současně příjem a vysílání dat a proto je potřeba využít poloduplexního režimu komunikace. U tohoto módu je potřeba využít algoritmus CSMA/CD (Carrier Sense Multiple Access / Collision Detection). Ten určuje pravidla, jak síťová zařízení reagují na kolizi. Kolize je stav, kdy se snaží 2 zařízení využívat datový kanál ve stejnou chvíli. Standardní sítě Ethernet využívají tuto službu CSMA/CD k fyzickému sledování provozu na trati na zúčastněných stanicích. Pokud v daném okamžiku nedochází k přenosu informace, může daná stanice vysílat. Jestliže se však pokouší vysílat datový rámec více stanic současně, vzniká kolize, která je detekována všemi zúčastněnými stanicemi. Po vzniku kolize se stanice snaží opakovat odesílání datového rámce po náhodném časovém intervalu. Pokud dojde k další kolizi, časový interval čekací doby se zvýší. Jednoduchým příkladem poloduplexního módu může být třeba komunikace skrz vysílačky [22], [27].

- **Plně duplexní režim**

V plně duplexním režimu komunikace mezi dvěma zařízeními může probíhat příjem i vysílání informací současně v jednom časovém okamžiku. Místo hubů se v této síti využívají switche (přepínače), které jsou s huby cenově srovnatelné. Skrz tyto switche se posílají pakety pouze počítačům, pro které jsou dané pakety určeny. Spojení mezi stanicemi musí být zprostředkováno např. pomocí kroucené dvojlinky nebo optického kabelu poskytujícího nezávislé vysílání a příjem dat. U kroucené dvojlinky slouží 2 páry

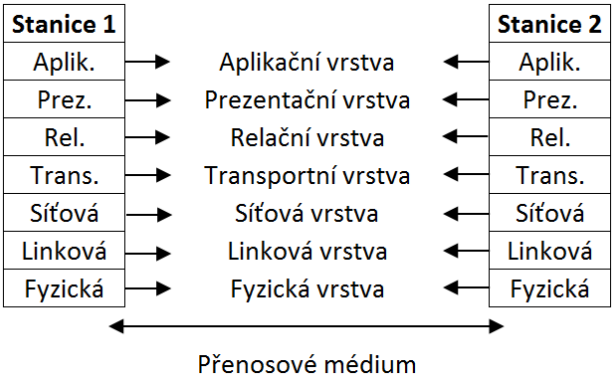
vodičů pro odesílání a 2 páry pro příjem. Switch si automaticky sám prohodí páry vodičů pro příjem za páry na odesílání, viz obrázek 4. TX v obrázku označuje vysílač (transceiver) a RX přijímač (receiver). Např. Fast Ethernet s kroucenou dvojlinkou poskytuje maximální šířku pásma 100 Mb/s. S využitím režimu full-duplex může stejný segment 100BASE-TX poskytnout celkovou šířku pásma 200 Mb/s. Nedochází zde tedy k potenciálním kolizím, což se projeví v efektivitě přenosu. Příkladem této komunikace je např. telefonní hovor [7], [21].



Obrázek 4: Překřížení vodičů – switch (vlastní)

1.3 Referenční model ISO/OSI

Referenční model ISO/OSI je koncepční model, který charakterizuje komunikační funkce telekomunikačního nebo výpočetního systému bez ohledu na jejich vnitřní strukturu a technologii. ISO/OSI byl vytvořen za účelem standardizace počítačových sítí. Tento model poskytuje základnu pro propojování systémů a vytvořené normy specifikují všeobecné principy sedmivrstvé architektury. Model ISO/OSI popisuje jednotlivé vrstvy a jejich funkce a slouží jako model pro programování součástí síťového subsystému v daných vrstvách, které mezi sebou komunikují. Komunikaci mezi dvěma stanicemi sedmi vrstvého modelu ISO/OSI znázorňuje následující obrázek (Obrázek 5) [20]. Nejnižší vrstvou modelu je vrstva fyzická. Za ní následuje vrstva linková, síťová a transportní. Nad transportní vrstvou jsou dále další 3 vrstvy, relační, prezentační a nejvyšší vrstvou modelu ISO/OSI je vrstva aplikační. Všechny tyto vrstvy jsou podrobněji rozepsány níže.



Obrázek 5: Model ISO/OSI (vlastní)

a) Fyzická vrstva

Nejnižší vrstvou referenčního modelu ISO/OSI je fyzická vrstva. Vrstva má za úkol přenášet bitový tok. Ten může být přenášen více způsoby. Nejčastějšími způsoby přenosu jsou elektrické impulsy nebo

optické signály. Propojení fyzických vrstev je zprostředkováno přenosovým médiem jako např. koaxiálním kabelem, kroucenou dvojlinkou nebo optickým kabelem. Na fyzické vrstvě je vytvořen tzv. fyzický okruh, na který jsou vkládána další zařízení jako modemy atd.

b) Linková vrstva

Následující vyšší vrstva je vrstva linková, která zajišťuje výměnu dat mezi stanicemi. Základním prvkem pro přenos je linkový rámec, skládající se ze záhlaví, dat a zápatí. Tento rámec je popsán podrobněji v kapitole 1.4.1.

c) Síťová vrstva

Třetí vrstvou v pořadí je vrstva síťová, jejíž úkolem je zajistit správnou adresaci dat. Základem je IP datagram, který se skládá ze záhlaví a datového pole. V síti WAN (Wide Area Network) je síťové rozhraní, zajišťující unikátní adresování dat. Tímto síťovým rozhraním se myslí např. síťová karta pro Ethernet [1], [20].

d) Transportní vrstva

Prostřední vrstvou v modelu ISO/OSI je transportní vrstva. Tato vrstva zajišťuje spojení a doručování informací mezi stanicemi. Nestará se vůbec o to, jak fungují nižší vrstvy, ale je na nich zcela závislá. Transportní pakety obsahují hlavičku a datovou část, a jsou přenášeny v datové části předcházející vrstvy.

e) Relační vrstva

Pátá vrstva je relační. Udržuje a koordinuje komunikaci. Přenáší data mezi aplikacemi, je softwarová.

f) Prezentační vrstva

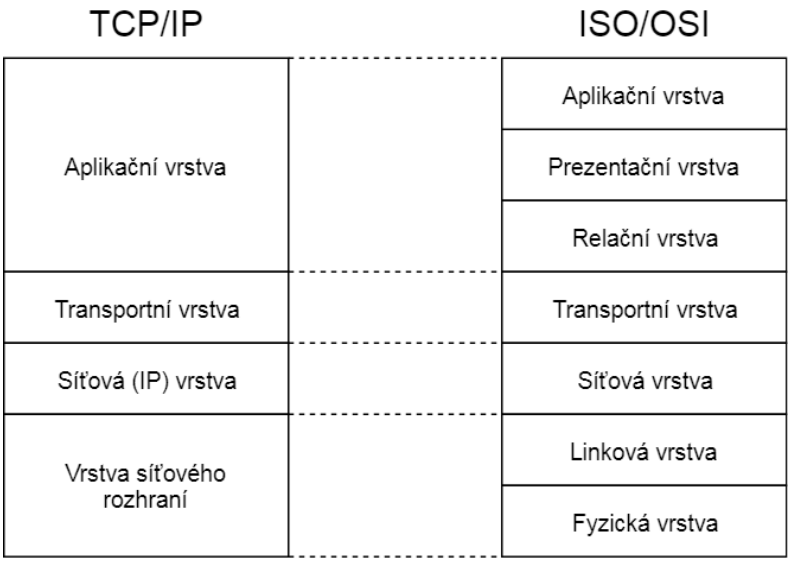
Předposlední vrstvou je vrstva prezentační, která má za úkol zajistit zabezpečení a reprezentaci dat. Zabezpečením se rozumí např. šifrování dat a digitální podpisy. Tato vrstva také řeší reprezentaci dat, která je na různých počítačích jiná. Např. zda mají přicházející data ve svých bytech MSB (Most Significant Bit) umístěn úplně vpravo nebo vlevo.

g) Aplikační vrstva

Aplikační vrstva je nejvyšší vrstvou modelu ISO/OSI, která určuje, jakým způsobem budou komunikovat aplikační programy se sítí. Příkladem těchto aplikačních programů může být například elektronická pošta nebo různé databáze atp. [1], [20].

1.4 Rodina protokolů TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) je rodina protokolů, která se převážně zabývá vrstvami vyššími, než je vrstva linková. Oproti referenčnímu modelu ISO/OSI rozlišuje TCP/IP pouze 4 vrstvy. Srovnání jednotlivých vrstev TCP/IP a ISO/OSI je znázorněno v následujícím obrázku (Obrázek 6). Za obrázkem byly stručně popsány jednotlivé vrstvy tohoto modelu.



Obrázek 6: TCP/IP – ISO/OSI (vlastní)

a) Vrstva síťového rozhraní

Nejnižší vrstvou u TCP/IP je vrstva síťového rozhraní, která zprostředkovává přístup k přenosovému médiu.

b) Síťová (IP) vrstva

Vyšší vrstvou je IP vrstva, která je srovnávána se síťovou vrstvou modelu ISO/OSI a jejím úkolem je přenášet IP pakety (datagramy) mezi stanicemi pomocí IP protokolů. Všechny tyto datagramy obsahují adresu příjemce, uloženou v záhlaví datagramu. Každé síťové rozhraní má svou jedinečnou IP adresu. To znamená, že jedno síťové rozhraní může mít více IP adres, avšak více síťových rozhraní nemůže mít stejnou IP adresu. Jednotlivé datagramy mohou být odesílány samostatně, přičemž mohou dorazit k adresátovi v jiném pořadí, než byly vyslány.

c) Transportní vrstva

Nad síťovou vrstvou se nachází transportní vrstva, často taky nazývaná jako TCP (Transmission Control Protocol) vrstva. Úkolem této vrstvy je realizovat spojení dvou koncových účastníků, tedy aplikačních programů, které běží na vzdálených stanicích. Toto spojení je zprostředkováno pomocí TCP a UDP (User Datagram Protocol) protokolu. Podle protokolu jsou data aplikacím dodávána pomocí TCP segmentů nebo UDP datagramů. Rozdílem mezi TCP a UDP protokolem je ten, že když jsou data (TCP segmenty nebo UDP datagramy) během přenosu ztraceny, TCP protokol zajistí opakovaní přenosu, protokol UDP nikoli. UDP protokol tedy nezajišťuje jakousi spolehlivost přenosu, ale je značně rychlejší.

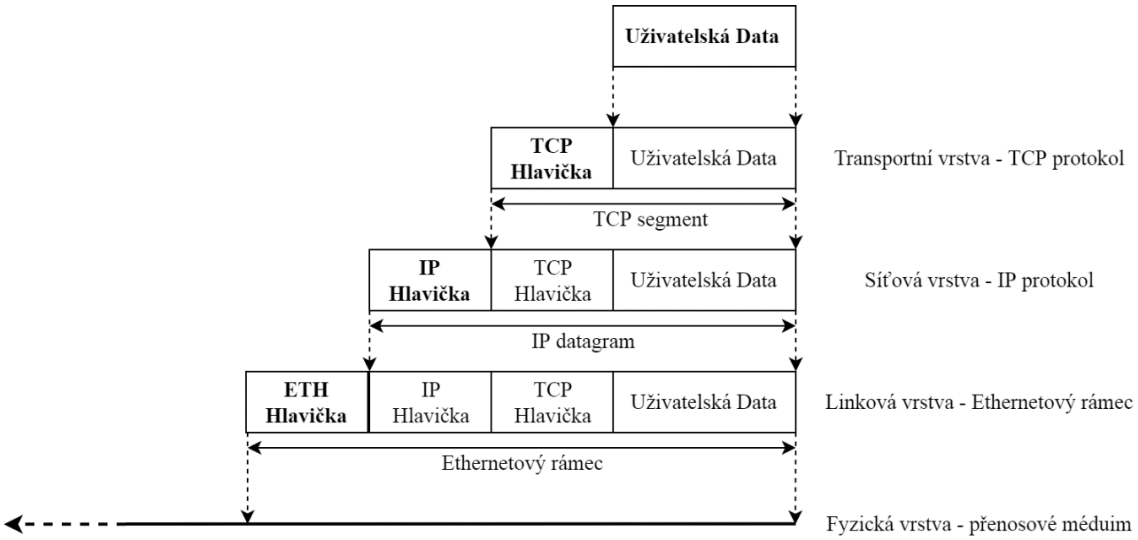
d) Aplikační vrstva

Nejvyšší vrstvou je vrstva aplikační. V této vrstvě se nacházejí samotné aplikační programy, komunikující přímo s TCP vrstvou [1], [5].

- **Struktura odesílaného paketu**

Na obrázku 7 je znázorněna struktura odesílaných dat po Ethernetové síti. Na úrovni aplikační vrstvy se nachází uživatelská data, která musí být odeslána. Na nižší vrstvě, tedy transportní je uplatněn TCP protokol a z tohoto důvodu je k uživatelským datům připojena hlavička TCP. Celému tomuto TCP

segmentu je přiřazena IP hlavička na úrovni síťové vrstvy. Tento blok dat se nazývá IP datagram, který je obsažen v datové části Ethernetového rámce. Ethernetový protokol si taktéž přiřadí svou hlavičku a poté jsou všechna tato data poslána po přenosovém médiu svému příjemci. Přenos znázorňuje šipka vlevo na úrovni fyzické vrstvy. Jakmile je paket adresátem přijat, pak stejným principem jsou tato data dekódována.



Obrázek 7: Struktura odesílaného paketu (vlastní)

V následujících podkapitolách jsou podrobněji rozebrány jednotlivé protokoly s předchozího obrázku. Jedná se o Ethernetový rámec, protokol IP a protokol TCP.

1.4.1 Ethernetový rámec

V této podkapitole je popsán formát Ethernetového rámce podle standardu IEEE 802.3, který je podrobně rozebrán v [2].

Paketem se rozumí formátovaný blok dat, který je přenášen po síti. V modelu ISO/OSI se řadí na úroveň linkové vrstvy a skládá se z řídících dat, jako např. hlavička a zakončení, a volitelných dat. Rámec je blok dat, který je obsahem paketu a je přenášen fyzické vrstvě (PHY) po přenosovém médiu. Formát rámce je popsán za pomocí oktetů neboli bytů (8 bitů). Ethernetový rámec popisuje obrázek 8, přičemž jednotlivé oktety jsou odesílány s hora dolů [6].

Obrázek 8: Ethernetový rámec (vlastní)

Jako první je odeslána preambule a Start Frame Delimeter, který svými posledními bity označuje začátek Ethernetového rámce. Následuje cílová a zdrojová MAC adresa, typ/délka a dalších 46 – 1500 oktetů se samotnými daty a případnou výplní. Poslední byty paketu tvoří kontrolní sekvence rámce a prodloužení. Všechny tyto části jsou zde podrobněji popsány.

- **Preambule**

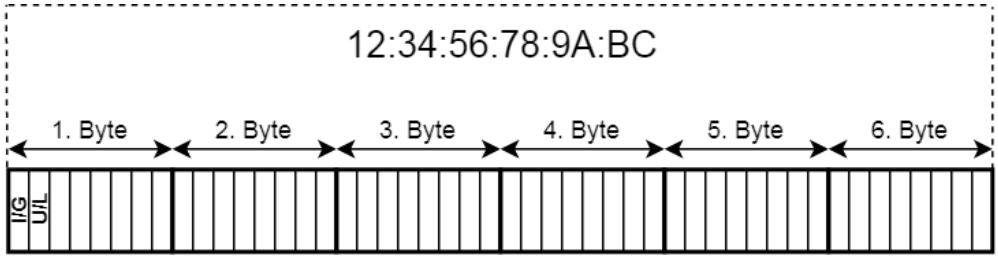
První částí paketu je preambule. Ta se skládá ze 7 oktetů ve formě střídajících se jedniček a nul "10101010" a slouží k synchronizaci hodinového signálu mezi vysílačem a přijímačem. U novějších verzí Ethernetu preambule postrádá význam, jelikož přijímač je v klidovém stavu neustále synchronizován s vysílačem. Pro zajištění kompatibility se staršími verzemi Ethernetu je však uchována [2].

- **SFD**

SFD neboli Start Frame Delimeter je jediný byte nabývající stejných hodnot jako oktety preambule s tím rozdílem, že je zakončený dvěma logickými jedničkami "10101011". Tyto dvě jedničky označují začátek MAC rámce [2].

- **Cílová MAC adresa**

Toto pole obsahuje 48 bitovou unikátní MAC adresu příjemce, pro kterého jsou data určena. Struktura MAC adresy je znázorněna na následujícím obrázku 9. Adresa využívá hexadecimální zápis ve tvaru 12:34:56:78:9A:BC, kde každý byte je oddělen dvojtečkou a přenášen od LSB (nejméně významného bitu) k MSB (nevýznamnějšímu bitu). Mezi speciální zápis patří adresa, která obsahuje samé logické '1' ve všech 6 oktetech a adresuje všechny stanice (broadcast). Její zápis tedy je FF:FF:FF:FF:FF:FF, avšak u zdrojové MAC adresy musí být I/G rovno log. '0' [1], [2], [19].



Obrázek 9: Struktura MAC adresy (vlastní)

První nejméně významný bit (dále jen LSB z angl. Least Significant Bit) v tomto poli označuje typ adresy pro identifikaci, zda jsou data určena pro skupinovou (multicast) nebo individuální (unicast) adresu. Druhý bit se používá k rozlišení mezi lokální nebo globální adresou, viz tabulka 2.

Tabulka 2: První 2 bity MAC adresy (vlastní)

I/G = 0 – Individuální adresa
I/G = 1 – Skupinová adresa
U/L = 0 – Globální adresa
U/L = 1 – Lokální adresa

- **Zdrojová MAC adresa**

Zdrojová MAC adresa označuje vysílací stanici MAC rámce. Pole má stejný formát jako cílová MAC adresa s tím rozdílem, že první bit I/G musí obsahovat logickou ‘0‘, protože vysílací stanice může mít pouze individuální adresu [1], [2].

- **Typ/Délka**

Pole Typ/Délka se skládá ze dvou oktetů a má dva významy. Může naznačovat typ paketu nebo jeho velikost.

- Pokud je hodnota tohoto pole v dekadickém tvaru menší nebo rovna 1500 (neboli 5DC hexadecimálně), pak pole Délka/Typ označuje velikost dat, která budou odeslána v následujícím poli *Data MAC*.
- Je-li hodnota tohoto pole v dekadickém tvaru větší nebo rovna 1536 (neboli 600 hexadecimálně), pak pole Délka / Typ označuje typ paketu vyšší vrstvy, který je v datové části MAC rámce [2].

- **Data MAC**

Datové pole MAC klienta obsahuje posloupnost oktetů, ve kterých přenášíme vlastní data. Implementace sítě Ethernet musí podporovat alespoň jednu ze tří maximálních velikostí. Jako základní rámec (basic frame) je označováno pole o velikosti 46 až 1500 oktetů. Mezi další dva patří Q-tagget rámec, který má maximální velikost 1504 oktetů a envelope rámec s velikostí do 1982 oktetů. Velikost se označuje parametrem MTU (Maximum Transmission Unit) [2].

- **Výplň**

Pokud velikost vlastních dat nedosahuje určitého minima, např. u základního rámce 46 oktetů, je třeba zajistit, aby byl rámec naplněn alespoň do své minimální velikosti. Tato výplň slouží zejména pro správné fungování algoritmu CSMA/CD. Po vlastních datech pak následuje kontrolní sekvence rámce [2].

- **Kontrolní sekvence rámce**

Frame Check Sequence field (FCS) neboli pole kontrolní sekvence obsahuje 4 byty. FCS obsahuje CRC (Cyclic Redundancy Check) hodnotu, což je hodnota, která je generována cyklickou redundantní kontrolou, která slouží pro kontrolu správnosti vyslaného/přijatého rámce. Kontroluje se celý rámec, tzn. cílová adresa, zdrojová adresa, typ/délka, data a výplň, mimo pole FCS samotné. Kódování je definováno generujícím polynomem (1).

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (1)$$

CRC hodnota je definována následujícím postupem:

a) Prvních 32 bitů rámce je komplementováno.
b) Hodnoty *n* bitů rámce jsou považovány za koeficienty polynomu *M(x)* stupně n – 1, kde první bit pole cílové adresy koresponduje s $x^{(n-1)}$ a poslední bit datového pole koresponduje s x^0, viz rovnice (1).
c) Polynom *M(x)* je vynásoben koeficientem x^{32} a vydělen polynomem *G(x)*.
d) Koeficienty *R(x)* se považují za 32 bitovou sekvenci, která vznikne dělením polynomů z předcházejícího bodu.
e) Sekvence bitů je komplementována a výsledkem je CRC.

- **Prodloužení**

Pole prodloužení navazuje na pole FCS a je tvořeno řadou prodlužovacích bitů, které se snadno odlišují od datových bitů. Využívá se u half-duplex režimu s přenosovou rychlostí 1 Gbit/s [2], [19].

1.4.2 IP - Internet Protokol

Jak už bylo zmíněno výše, IP protokol zajišťuje přenos dat mezi dvěma počítači v internetové síti, neboli spojení jednotlivých lokálních sítí. Všechna síťová rozhraní mají svůj jedinečný identifikátor, unikátní IP adresu, která slouží pro adresaci odeslaných dat. IP adresy jsou neseny v datagramech, putujících ke svému příjemci nezávisle na sobě. Mezi základní internetové protokoly se řadí IPv4, IPv6, ICMP a IGMP[4].

- **IPv4**

Internet Protocol verze 4 je momentálně jedním z nejrozšířenějších protokolů v Ethernetové komunikaci i přesto, že tento protokol sám o sobě nemá schopnost spolehlivého přenosu dat. Tuto schopnost za něj řeší TCP protokol z vyšší vrstvy (viz podkapitola 1.4.3), který vyžaduje opětovné odeslání paketu v případě jeho poškození. Pokud chtějí spolu komunikovat dvě zařízení na úrovni IP vrstvy, pak musí být tato data zapouzdřena v datovém poli MAC rámce na úrovni linkové vrstvy, viz. kapitola 1.4.1.

Správu a přidělování IP adres má na starost organizace IANA (z angl. Internet Assigned Numbers Authority). U této verze má IP adresa velikost 32 bitů. To znamená, že IPv4 je schopen adresovat až 2^{32} adres, tzn. přes 4 miliardy potenciálních adresátů. V důsledku vyčerpání všech adres byla vyvinuta verze IPv6, která je schopna teoreticky pojmou až 2^{128} unikátních adres [18].

Hlavička IP datagramu obsahuje celkem 14 datových polí, viz obrázek 10. Minimální velikost hlavičky je 20 bytů a údaje jsou do ní zapisovány pomocí konvence Big Endian, což znamená, že se zápis začíná od MSB. Jednotlivá datová pole hlavičky IP datagramu jsou blíže popsána za následujícím obrázkem [1], [4].

Offset	Oktet 0								Oktet 1								Oktet 2								Oktet 3							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Version				IHL				Type of Service								Total length															
4	Identification																Flags			Fragment offset											
8	Time To Live								Protocol								Header Checksum															
12	Source Adress																															
16	Destination Adress																															
20…	Options																								Padding							

Obrázek 10: Hlavička IPv4 datagramu (vlastní podle [4])

- **Version:** Verze protokolu IPv4 = 4 bity
- **IHL (Internet Header Length):** Definuje velikost hlavičky v 32 bitových slovech. Typickou a zároveň minimální velikostí správného záhlaví je 20 bytů tzn. IHL = 5.
- **Type of Service:** Toto pole se ve většině případů vyplňuje samými nulami, jelikož v praxi nenašlo své uplatnění. Slouží hlavně pro potřebu podpory služeb QoS.
- **Total Length:** Velikostí tohoto datového pole jsou 2 oktety (byty), které označují celkovou velikost IP datagramu v bytech včetně hlavičky.
- **Identification:** Pole sloužící k identifikaci IP datagramů, v případě fragmentace. Pokud k fragmentaci nedochází, je toto pole zbytečné.

- **Flags:** Příznaky určují, zda může být paket fragmentován či nikoli. Má velikost 3 bitů.
- **Fragment offset:** Jeli původní paket fragmentován, pak toto pole uchovává přesné pozice jednotlivých fragmentů.
- **Time to Live:** Hodnota, která udává maximální počet průchodů paketu směrovači (routery). Při každém průchodu směrovačem se tato hodnota dekrementuje o jedničku. Jakmile hodnota TTL klesne na nulu, pak je paket zahozen. Tímto způsobem je síť ošetřena před zacyklením paketů mezi routery.
- **Protocol:** Bytového pole určuje protokol vyšší vrstvy, který využívá IP datagram ke svému transportu. Nejčastěji se objevují protokoly TCP a UDP, ale využívají se i služební protokoly IGMP a ICMP, které jsou formálně součástí síťové vrstvy s tím, že se tváří jako protokoly vrstvy vyšší. Organizace IANA má na starost správu čísel jednotlivých protokolů. Čísla nejvíce používaných protokolů jsou uvedena v tabulce 3.

Tabulka 3: Přiřazení čísel protokolům vyšší vrstvy

Číslo protokolu vyšší vrstvy	Protokol
1_{10}	ICMP
2_{10}	IGMP
6_{10}	TCP
17_{10}	UDP

- **Header Checksum:** Kontrolní součet je prováděn pouze z hlavičky nikoli tedy z celého IP datagramu.
- **Source Address:** Předposlední datové pole hlavičky IP datagramu. Je to 32 bitová IPv4 adresa odesílatele.
- **Destination Address:** Poslední datové pole hlavičky IP datagramu. Je to 32 bitová IPv4 adresa příjemce.
- **Options:** Volitelné položky jsou používány jen zřídka a mívají různou velikost.
- **Padding:** Doplňuje volitelné položky tak, aby hlavička byla zarovnána na celé násobky 32 bitů, z důvodu, aby se dokázala do IHL zapsat velikost hlavičky.

Za hlavičkou IP datagramu následuje datová část, která obsahuje protokoly a data vyšších vrstev. Pakety TCP a UDP pak obsahují pole pro port (zdrojový i cílový), který specifikuje, pro kterou aplikační vrstvu jsou data určena.

Celý tento IP datagram je zabalen do datové části MAC rámce, který má maximální velikost danou parametrem MTU (nejčastěji 1500 bytů). Pro přesnou adresaci příjemce musíme znát nejen IP adresu, ale také cílovou MAC adresu.

1.4.3 TCP – Transmission Control Protocol

Na úrovni transportní vrstvy v rodině protokolů TCP/IP je možné si zvolit ze dvou protokolů. Konkrétně se jedná o protokol TCP a UDP. Nespolehlivý, ale za to rychlejší přenos dat je zprostředkován protokolem UDP, který ve své podstatě pouze zprostředkovává přístup aplikační vrstvě. V této práci však bylo využito protokolu TCP, který zajišťuje spolehlivost přenosu, ale není rychlý jako protokol

UDP. Dalším rozdílem mezi protokolem TCP a UDP je ten, že TCP protokol nabízí své služby jako connection-oriented, což znamená, že před jakoukoliv výměnou dat musí být prvně navázáno spojení.

K zajištění přenosu dat využívá protokol TCP tzv. kladného potvrzování (z angl. positive acknowledgement). Tento typ potvrzování funguje tím způsobem, že odesílatel odešle blok dat (přesněji TCP segment, viz obrázek 7), a příjemce po převzetí dat informuje odesílatele o tom, že data přijal. Pokud se však k odesílateli potvrzení nedostaví do určitého časového limitu (timeout), odesílatel následně čeká náhodnou dobu, po které se pokouší data odeslat znovu.

Tak jako každý komunikační protokol, svou hlavičku má i protokol TCP. Tato hlavička má v základním rozsahu velikost minimálně 20 Bytů a její znázornění je zobrazeno na obrázku 11.

Offset	Oktet 0								Oktet 1								Oktet 2								Oktet 3							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source Port																Destination Port															
4	Sequence																															
8	Acknowledgement																															
12	HLEN				Reserved								Flags				Window															
16	Checksum																Urgent Pointer															
20…	Options																								Padding							

Obrázek 11: Hlavička TCP protokolu (vlastní podle [4])

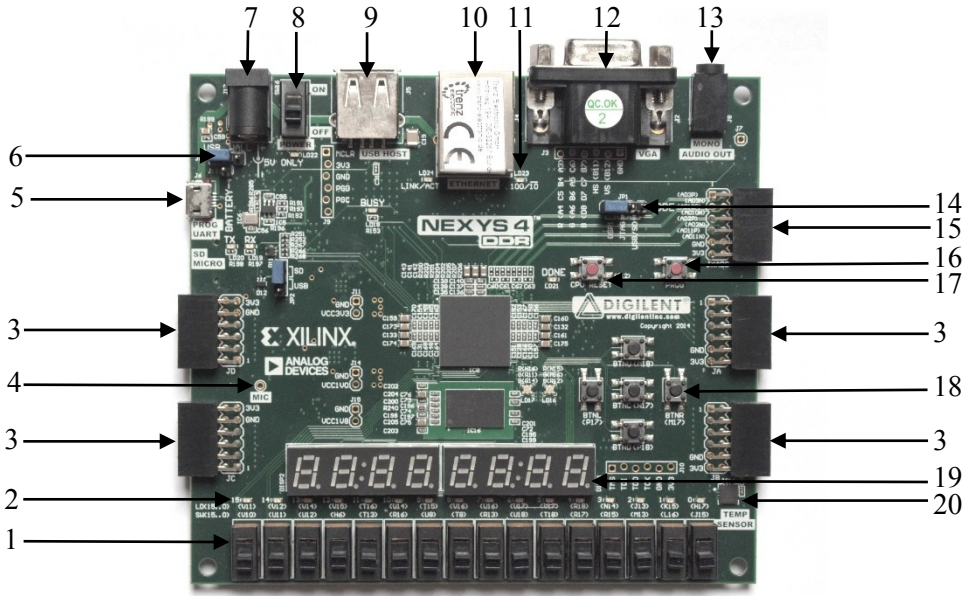
V následujících bodech je opět popsána funkce jednotlivých datových polí.

- **Source port:** Položka označující zdrojový port.
- **Destination Port:** Datová část, která určuje, jakému portu jsou data určena.
- **Sequence:** Číslo udávající pořadí přeneseného Bytu, pro zajištění potvrzování příjmu dat.
- **Acknowledgement:** Číslo, které odešle příjemce odesílateli po obdržení dat.
- **HLEN:** Velikost hlavičky.
- **Reserved:** Toto pole je vyhrazeno pro nějaké budoucí využití, a proto se nastavuje na samé nuly.
- **Flags:** TCP protokol umožňuje tzv. „předbíhání“. Jestliže se v segmentu nacházejí naléhavá data (urgent data), pak je tato informace označena v tomto datovém poli.
- **Window:** Označuje velikost datového okénka, které příjemce poskytuje odesílateli. Respektive se jedná o velikost vyrovnávacího paměti, kterou má příjemce k dispozici.
- **Checksum:** Kontrolní součet velikosti hlavičky.
- **Urgent Pointer:** Ukazatel na konec naléhavých dat v segmentu.
- **Options:** Volitelné položky jsou používány jen zřídka a mívají různou velikost.
- **Padding:** Doplňuje volitelné položky tak, aby hlavička byla zarovnána na celé násobky 32 bitů.

2 Vývojová deska FPGA Digilent Nexys 4 DDR Artix-7™

Pro komunikaci Ethernetovým rozhraním v této práci byla použita deska FPGA Nexys 4 DDR Artix-7™ (dále jen Nexys 4) firmy Digilent, Inc. Tato platforma je založena na programovatelném hradlovém poli (dále jen FPGA) vytvořeným společností Xilinx®.

Jádrem této desky je čip FPGA Artix-7™, ke kterému vedou různé komunikační rozhraní a další periferie. Deska zahrnuje 16 spínačů, nabývajících hodnot logické 0 a logické 1, 16 LED diod, dva sedmi segmentové displeje, tlačítka, teplotní senzor, 3osý akcelerometr, mikrofon, reproduktor a slot na MicroSD kartu. Komunikace s jinými zařízeními je zprostředkována mnoha komunikačními konektory, mezi které patří například USB (Universal Serial Bus), Ethernet, audio jack, nebo VGA (Video Graphics Array). Náhled desky s popsanými periferiemi je zobrazen na obrázku 12 [3].



Obrázek 12: FPGA Digilent Nexys 4 DDR (vlastní)

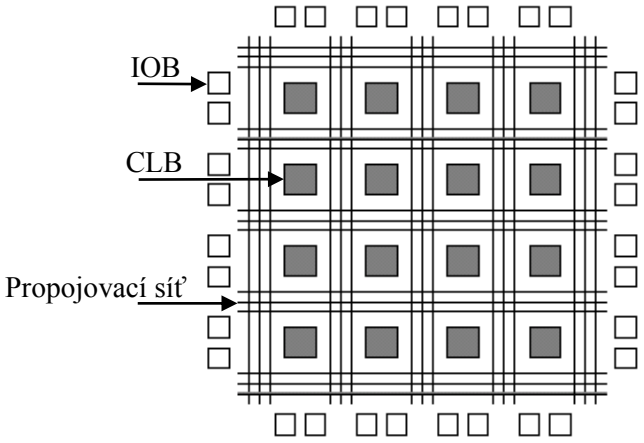
S předcházejícím obrázkem souvisí následující tabulka 4, která přehledně popisuje periferie FPGA desky NEXYS 4 DDR.

Tabulka 4: Popis desky Digilent Nexys 4 DDR (vlastní)

Pozice	Komponenta	Pozice	Komponenta
1	Spínače	**11**	LED dioda - typ Eth. přenosu
2	LED diody	**12**	VGA konektor
3	Pmod konektory	**13**	Audio konektor
4	Mikrofon	**14**	Jumper - programovací mód
5	USB (JTAG, UART)	**15**	Pmod konektor - analog
6	Jumper s výběrem napájení	**16**	Tlačítko - FPGA restart
7	Napájecí konektor	**17**	Tlačítko - CPU restart
8	Spínač napájení	**18**	Tlačítka
9	USB konektor	**19**	7 segmentové displeje
10	Ethernetový konektor	**20**	Teploměr

2.1 Programovatelná hradlová pole FPGA

Programovatelná hradlová pole FPGA (z angl. Field Programmable Gate Array) jsou logické integrované obvody, které se skládají z velkého množství elementárních logických prvků. Tyto prvky jsou zpravidla umístěny do maticové struktury tak, aby mohly být vzájemně propojovány. Toto propojování je realizováno za pomoci konfigurovatelné propojovací sítě přímo na čipu FPGA. Základní architektura FPGA je znázorněna na obrázku 13.



Obrázek 13: Struktura FPGA

Základními logickými prvky FPGA jsou tzv. konfigurovatelné logické bloky CLB (Configurable Logic Blocks), které jsou uspořádány do maticové struktury. Na okrajích čipu jsou vstupně/výstupní bloky IOB (Input/Output Blocks). Jak již bylo zmíněno, všechny jednotlivé prvky jsou propojovány hustou sítí propojovacích vodičů tzv. Routing Channels. FPGA také obsahují další prvky jako např. statické paměti Block RAM [17].

2.2 Možnosti připojení rozhraní Ethernet k programovatelné logice

Připojení Ethernetového rozhraní k programovatelné logice je možné za pomoci dvou způsobů. Prvním způsobem je připojení IP periferií, které pracují s integrovaným Ethernetovým portem, jako části procesoru MicroBlaze. Druhou možností je propojení samostatného Ethernetového kontroléru s konektorem Pmod na desce NEXYS 4, viz kapitola 2.2.2.

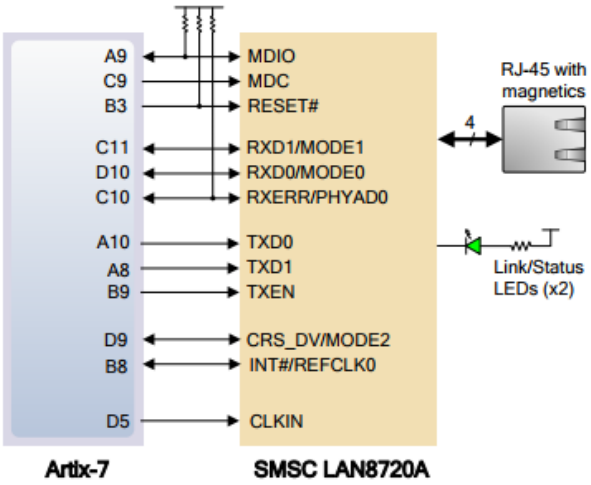
2.2.1 Ethernet PHY

FPGA čip Artix-7 na desce Nexys 4 je propojen s Ethernet PHY (fyzickou vrstvou LAN8720A), která je rovněž propojena s konektorem Ethernetu RJ-45 (Obrázek 14). Fyzická vrstva využívá rozhraní RMII (Reduced Media Independent Interface) a podporuje typy Ethernetu 10BASE-T a 100BASE-TX. Z těchto typů je zřejmá přenosová rychlost, která je 10 Mbit/s pro 10BASE-T a 100 Mbit/s pro 100Base-TX. Pro tyto dva typy Ethernetu také platí standardizace IEEE 802.3.

Dvě vestavěné LED diody na desce, konkrétně dioda LD23 = LED2, LD24 = LED1, jsou spojeny s fyzickou vrstvou a poskytují informaci o nastavené přenosové rychlosti a stavu.

Po spuštění desky FPGA nebo po jejím restartu se nastaví PHY do defaultního nastavení [3]:

- Rozhraní RMII,
- Rychlost 10/100 Mbit/s,
- PHY adresa = 00001.



Obrázek 14: Propojení Artix-7 a PHY Ethernetu [3]

a) Rozhraní MII

Rozhraní MII (Media Independent Interface) je standardním rozhraním, které spojuje fyzickou vrstvu s linkovou vrstvou za podpory přenosové rychlosti 10 Mbit/s a 100 Mbit/s. Využívá se zde přenos pro vysílání i příjem dat po tzv. nibblech (4 bitech). Rozhraní může být použito pro různé druhy přenosových médií jako např. stíněných i nestíněných metalických kabelů nebo také optických kabelů a dalších.

b) Rozhraní RMII

Rozhraní RMII (Reduced Media Independent Interface) je redukovaná verze rozhraní MII. Používá se pro propojení s vrstvou Ethernet PHY podle standardu IEEE 802.3. RMII se používá z důvodu ušetření potřebných pinů a pro připojení více fyzických vrstev u zařízení, která potřebují více Ethernetových portů.

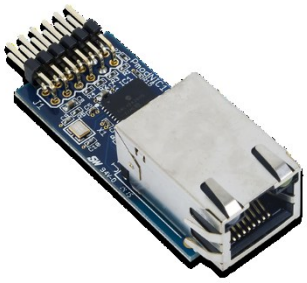
Oproti MII se zde využívá přenos vyslaných a přijatých dat pouze po 2 bitech. Podporovaná rychlost přenosu je však stejná jako u MII rozhraní. Dalším rozdílem je, že dva hodinové signály u MII (TX_CLK a RX_CLK) jsou v rozhraní RMII nahrazeny jedním společným hodinovým signálem REF_CLK. Vzhledem k situaci, že RMII má menší šířku datové sběrnice a podporuje stejné rychlosti přenosu jako MII, je zde navýšená taktovací frekvence hodinového signálu na dvojnásobek, tzn. na 50 MHz. Přehled jednotlivých signálů, jejich směru, označení a šířky je uveden v tabulce 5.

Tabulka 5: Signály RMII rozhraní

Signál	Směr (od MAC)	Šířka	Popis
REF_CLK	obousměrný	1	Hodinový signál pro PHY
TXD	výstup	2	Data pro odeslání
TX_EN	výstup	1	Povolení odesílání
RXD	vstup	2	Data pro příjem
CRS_DV	vstup	1	Data na RXD + detekce nosné
RX_ER	vstup	1	Chyba příjmu dat
MDC	výstup	1	MIIM - hodinový signál
MDIO	obousměrný	1	MIIM - data

2.2.2 Pmod NIC100: Network Interface Controller

Druhou metodou připojení Ethernetového rozhraní k desce Nexys 4 je využití modulu Network Interface Controller 100. Modul Pmod NIC100 je navržen tak, aby poskytoval plnohodnotné Ethernetové rozhraní za pomoci samostatného Ethernetového kontroléru Microchip® ENC424J600. Zařízení ENC424J600 poskytuje integrovanou podporu vrstev MAC a PHY a je připojeno do Pmod konektoru (Obrázek 15). Funkce Ethernet tedy může být přidána do libovolné systémové desky, prostřednictvím protokolu SPI [8].



Obrázek 15: Pmod NIC100 [8]

Modul NIC100 představuje:

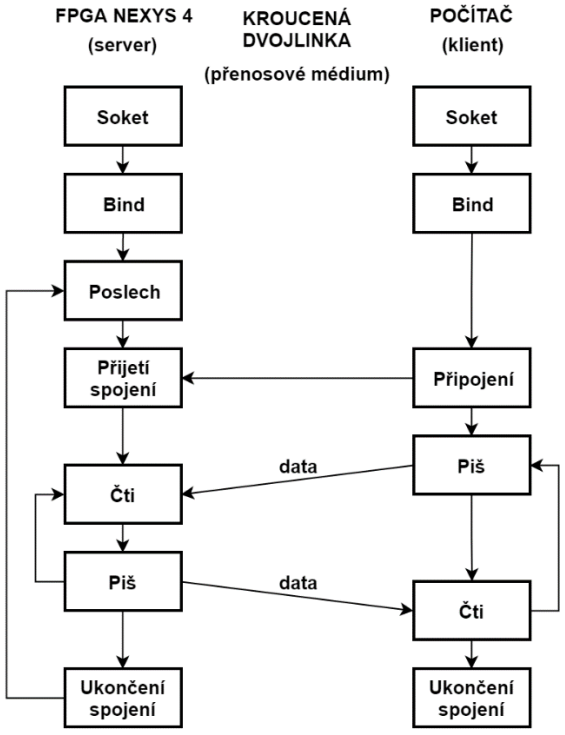
- Standard IEEE 802.3, kompatibilním ethernetovým řadičem,
- Podporu MAC a PHY vrstvy,
- Podporu 10BASE-T a 100BASE-TX,
- Přenosovou rychlost 10/100 Mbit/s,
- 12 pinový Pmod konektor s rozhraním SPI,
- Konektor RJ-45.

3 Demonstrační úloha

V této kapitole je popsán návrh a implementace logického obvodu demonstrační úlohy a její softwarového řešení. První část samotné realizace je provedena ve vývojovém prostředí Vivado s využitím vestavěného procesorového jádra MicroBlaze na desce Nexys 4. V druhé části je využito kompletního open source TCP/IP stacku lwIP, který slouží pro realizaci protokolů IP vrstvy a transportní vrstvy. Do těchto vrstev se řadí protokoly IPv4, ICMP, IGMP, TCP a UDP. Ovládání rozhraní s využitým procesorem MicroBlaze je realizováno v programu Software Development Kit (dále jen SDK) v jazyce C.

3.1 Návrh demonstrační úlohy

Cílem této práce bylo vytvořit demonstrační úlohu, která bude zajišťovat přenos dat mezi vývojovou deskou FPGA Nexys 4 a počítačem za pomoci Ethernetového rozhraní, viz obrázek 16. Z obrázku je patrné, že na přípravku Nexys 4 bude běžet server, na který se bude připojovat vzdálený počítač (klient), pomocí terminálu. Propojení těchto dvou stanic bude zprostředkováno pomocí přenosového média, konkrétně kroucené dvojlinky zakončené konektorem RJ-45. Po vytvoření spojení mezi těmito dvěma stanicemi je možné přenášet data, která budou v serveru přečtena a zpracována, a na jejich základě server vykoná daný příkaz.

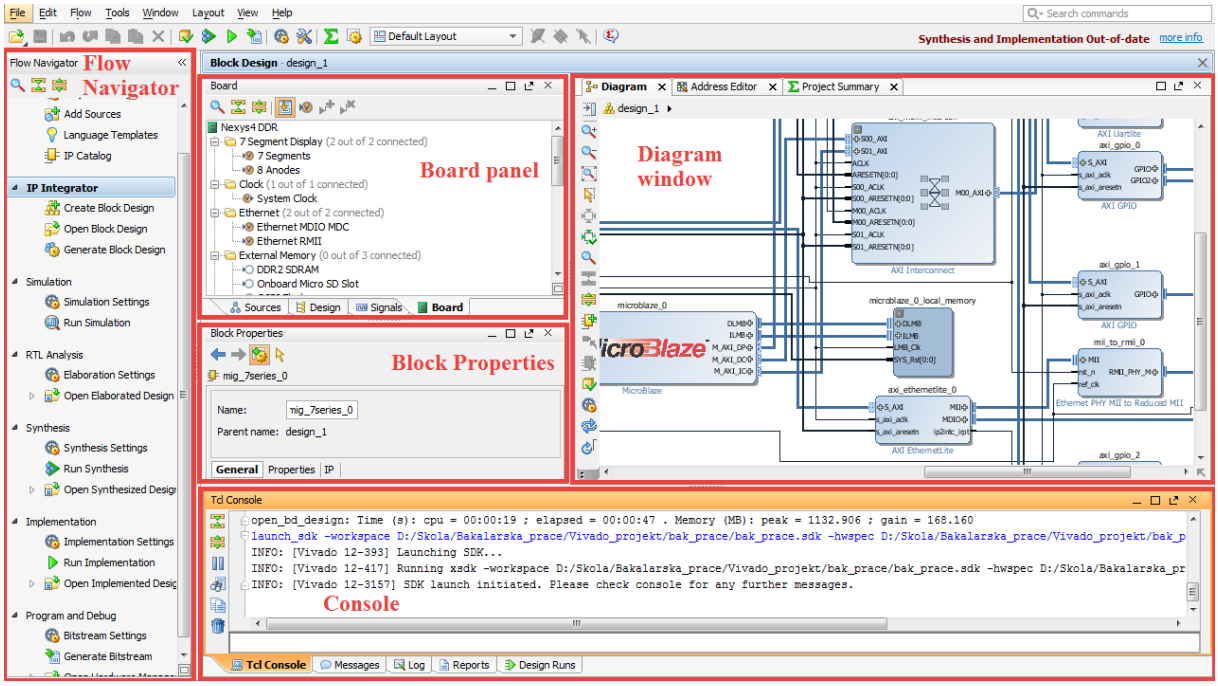


Obrázek 16: Návrh demonstrační úlohy

Pro demonstraci správně přenesených dat budou využity periferie desky Nexys 4. Server bude reagovat na dvě klíčová slova. Klíčové slovo „Switch“ a „Temp: “ s následující dvoucifernou hodnotou (hodnota představuje teplotu ve stupních Celsia). Dorazí-li „Switch“, pak server načte hodnotu přepínačů a odešle ji klientovi. Dorazí-li např. „Temp: 36“, zobrazí se na 7 segmentovém displeji desky Nexys 4: „36°C“. Pokud serveru dorazí jiná data, než výše uvedená klíčová slova, server odešle odesílateli zprávu o tom, aby některé z těchto klíčových slov zadal. Přesný popis jednotlivých bloků z předcházejícího obrázku je uveden v 3.3.1.

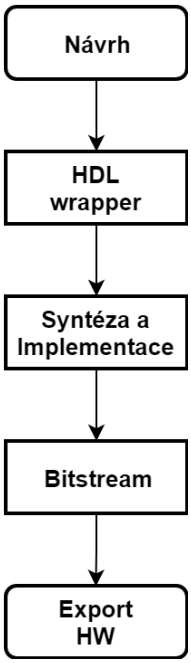
3.2 Vývojové prostředí Vivado

Ve vývojovém prostředí Vivado, byl vytvořen RTL projekt, který umožňuje návrh blokových schémat za pomoci IP (Intellectual Property) bloků, až po jejich implementaci. Návrh se dá realizovat také pomocí jazyka VHDL nebo Verilog. Na obrázku 17 je zobrazeno hlavní okno tohoto prostředí. Vlevo je umístěn Flow Navigator panel, který nabízí různé možnosti, jak vytvořit návrh hardwaru, provést jeho simulaci, spustit syntézu a implementaci a generovat bitový soubor tzv. bit file. V tomto řešení bylo využito prvně IP Integratoru pro vytvoření blokového schématu z IP bloků, viz Diagram window v obrázku 17. Uprostřed grafického rozhraní, v okně board panel jsou informace o aktuálně využívaných periferiích desky Nexys 4. Pod tímto oknem je panel Block Properties, který poskytuje informace o vlastnostech jednotlivých IP bloků.



Obrázek 17: Hlavní okno vývojového prostředí Vivado

Při vytváření logického obvodu v tomto vývojovém prostředí bylo postupováno podle následujícího diagramu (Obrázek 18). Prvně bylo zapotřebí vytvořit blokové schéma obvodu. Následovalo vytvoření tzv. HDL wrapperu, syntéza, implementace, vygenerování konfiguračního souboru .bit a export do vývojového prostředí SDK. Podrobnější popis jednotlivých bloků je uveden v následující podkapitole 3.2.1.



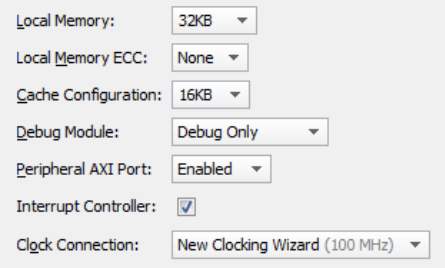
Obrázek 18: Diagram pracovního postupu při návrhu logického obvodu

3.2.1 Realizace logického obvodu

Hlavním blokem schématu z knihovny prvků je vestavěné (embedded) procesorové jádro, reprezentováno IP blokem Xilinx Microblaze, zobrazeným na následujícím obrázku 19. Jedná se o takzvaný soft-core processor, který byl sestaven z logických prvků. Tento prvek byl následně naprogramován v jazyce C v SDK a řídí Ethernetovou komunikaci (viz SW řešení - kapitola 3.3).

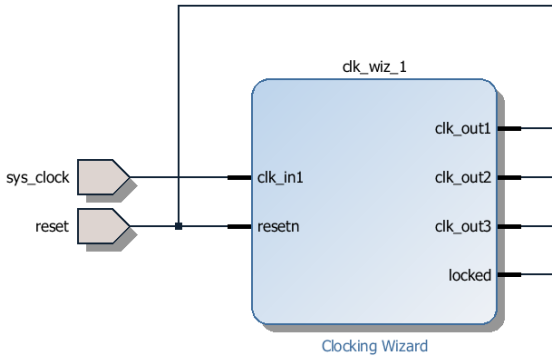
Obrázek 19: IP blok MicroBlaze

Po přidání tohoto bloku do designu byly nastaveny jeho parametry. Připojení bloku MicroBlaze generuje lokální paměť vybrané velikosti a lze konfigurovat mezipaměti. MicroBlaze debug module, periferní AXI port, možnost přerušení a zdroj hodin jsou přidány a propojeny také, viz obrázek 20. Na základě tohoto nastavení se v blokovém schématu automaticky vytvoří další IP bloky, které jsou přidány do našeho hardwarového designu.



Obrázek 20: Nastavení parametrů IP bloku MicroBlaze

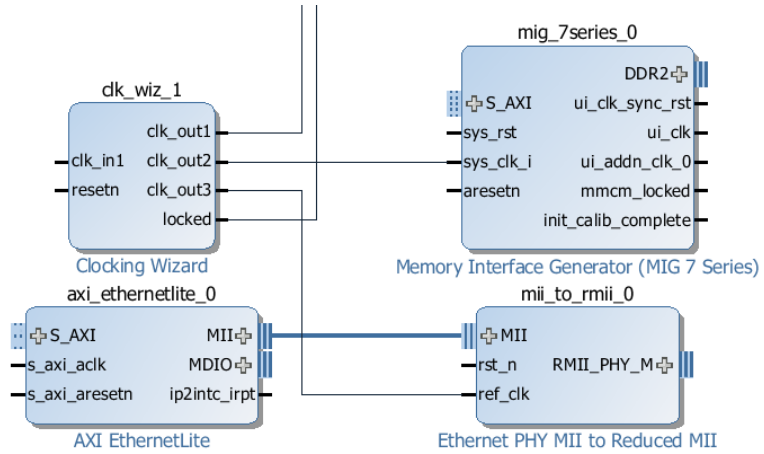
Jedním z těchto automaticky přidaných bloků je IP Clock Wizard, do kterého vstupuje 100 MHz hodinový signál sys_clock, buzený z krystalu na desce Nexys 4, viz obrázek 21. Druhým vstupem do tohoto bloku je port reset, který je propojen s resetovacím tlačítkem na desce. Úkolem tohoto bloku je vytvořit další potřebné výstupní hodiny. Konkrétně výstup clk_out2 s 200 MHz frekvencí a clk_out3 s taktovací frekvencí 50 MHz. Mezi další vygenerované bloky se řadí sběrnice, různé resetovací obvody atd.



Obrázek 21: IP Blok Clock Wizard se vstupními porty a výstupními hodinovými signály

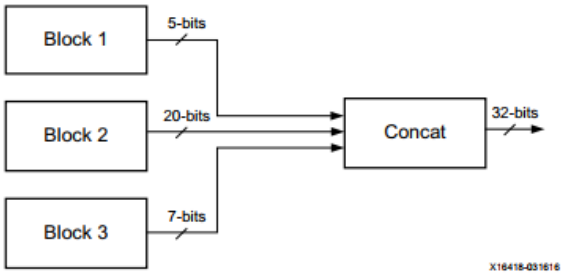
V následujícím kroku bylo potřeba přidat dalších 7 IP bloků, konkrétně AXI Ethernet Lite, Ethernet PHY MII to reduced MII, AXI Uartlite, AXI Timer, dva bloky AXI GPIO a Memory Interface Generator. Nejdůležitější bloky logického obvodu jsou stručně popsány v podkapitole 3.2.2.

Po přidání jednotlivých bloků bylo zapotřebí jejich manuální propojení. Jako první byl propojen výstup MII, IP bloku AXI EthernetLite, se vstupem MII, bloku MII to RMII. Vytvořený hodinový výstup clk_out2 (200 MHz) v IP Clock Wizard byl propojen se vstupem sys_clk_i bloku Memory Interface Generator a clk_out3 (50 MHz) se vstupem ref_clk bloku MII to RMII, viz obrázek 22. Následné vytvoření dalších cest mezi bloky bylo realizováno automaticky za pomoci funkce Run Connection Automation ve Vivadu.



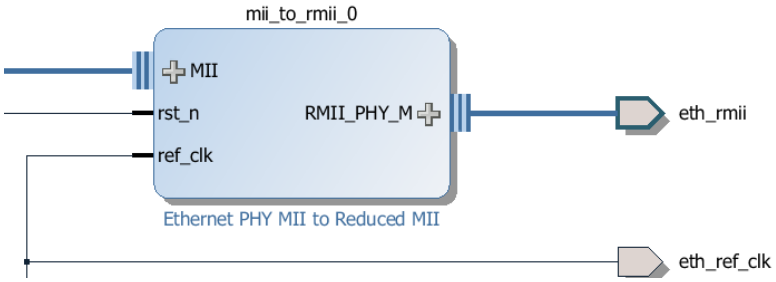
Obrázek 22: Propojení potřebných hodinových signálů pro bloky MII to RMII a MIG

Výstupní signály sloužící pro přerušení, z bloků AXI Timer a AXI EthernetLite, bylo potřeba sjednotit. K tomu bylo využito IP jádro Concat, které umožňuje zřetězit přicházející signály různých šířek do jediné společné linky (Obrázek 23).



Obrázek 23: Funkce bloku Concat

V této chvíli bylo třeba vytvořit výstupní hodinový port s názvem eth_ref_clk s frekvencí 50 MHz. Tento hodinový signál pochází také z výstupu clk_out3 z bloku Clock Wizard, stejně jako signál vstupující do bloku MII to RMII, viz obrázek 22 a obrázek 24.



Obrázek 24: Vytvoření portu eth_ref_clk a jeho propojení

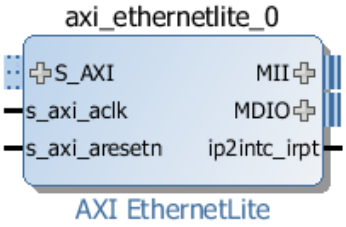
Posledním krokem při návrhu logického obvodu bylo nutné propojit výše vytvořený pin eth_ref_clk se správným fyzickým pinem na FPGA za pomoci vytvoření nového souboru .XDC, ve kterém je toto propojení popsáno. Finální schéma logického obvodu je uvedeno v příloze 0 na konci dokumentu a také v elektronické příloze ve formátu pdf.

3.2.2 Využité IP bloky

Tato podkapitola obsahuje stručný popis nejpodstatnějších IP bloků, které byly v návrhu využity. Kompletní dokumentace k jednotlivým blokům bude vždy uvedena zdrojem v následujících odstavcích.

- **AXI Ethernet Lite**

IP blok AXI Ethernet Lite Media Access Controller (MAC) je navržen tak, aby zahrnoval příslušné funkce popsané v normě IEEE Std. 802.3 Media Independent Interface (MII). Návrh podporuje rozhraní o rychlosti přenosu 10 Mbit/s a 100 Mbit/s, známé taky jako Fast Ethernet. Komunikuje s procesorem pomocí rozhraní AXI4 nebo AXI4-Lite a má schopnost přerušení při vysílání a příjmu. Obsahuje volitelné MDIO (Management Data Input/Output) rozhraní pro přístup k PHY.



Obrázek 25: Blok AXI Ethernet Lite

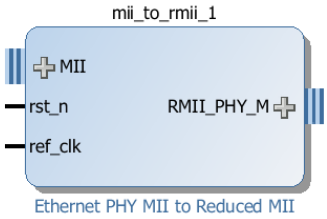
Tento blok se skládá ze základních šesti částí, mezi které patří rozhraní AXI4-Lite, TX Buffer, RX Buffer, Transmit, Receiver a MDIO Master Interface. Jednotlivé části jsou zde popsány.

- **Rozhraní AXI4-Lite**
  Tato část bloku AXI Ethernet-Lite poskytuje podřadné AXI4-Lite rozhraní, sloužící pro přístup k registru a přenosu dat.
- **TX Buffer**
  Část TX Buffer je vyrovnávací paměť o velikosti 2 kB, sloužící pro odesílaná data. Tato vyrovnávací paměť uchovává odesílaná data do doby, dokud z nich nebude vytvořen kompletní Ethernetový rámec, viz kapitola 1.4.1.
- **RX Buffer**
  RX Buffer má stejnou funkci i velikost jako TX Buffer, s tím rozdílem, že slouží pro přijatá data. Tato vyrovnávací paměť uchovává pakety do doby, než bude proveden kontrolní součet. Obě vyrovnávací paměti mohou být volitelně rozšířeny o velikost dalších 2 kB.
- **Transmit**
  Tento blok Transmit (Vysílač) se skládá z dalších částí jako je modul generující CRC, multiplexor, registr TX FIFO (First In First Out) a modul vysílacího rozhraní. V tomto bloku je také řídící jednotka multiplexoru, která si organizuje pořadí datových polí Ethernetového rámce, který následně odesílá do registru FIFO. Mezi datová pole patří preambule, SFD a další (viz kapitola 1.4.1). V okamžiku, kdy je rámec odeslán na fyzickou vrstvu (PHY), je generováno přerušení a aktualizován vysílací řídící registr.
- **Receiver**
  Blok Receiver (Přijímač) se skládá z modulu přijímacího rozhraní, kontroly zpětné vazby multiplexoru (loopback MUX), registru RX FIFO, bloku kontrolujícího CRC a modulu řídícího příjem dat. Přicházející data z fyzické vrstvy prochází kontrolou zpětné vazby multiplexoru a jsou uložena do registru RX FIFO. Poté data projdou blokem kontrolující CRC, a pokud tato hodnota souhlasí s hodnotou uloženou v příchozím Ethernetovém rámci, je generováno přerušení.
- **MDIO Interface**
  Rozhraní MDIO umožňuje správu a přístup k fyzické vrstvě.

Podrobnější informace o bloku AXI Ethernet Lite, včetně blokového schématu jeho vnitřní struktury a vnitřních signálů, jsou uvedeny v [10]. Vzhledem k tomu, že fyzická vrstva dokáže využívat pouze rozhraní RMII (viz předcházející podkapitola 2.2.1), musí být za blok AXI Ethernet Lite zapojeno další IP jádro, a to konkrétně MII to RMII.

- **Ethernet PHY MII to Reduced MII**

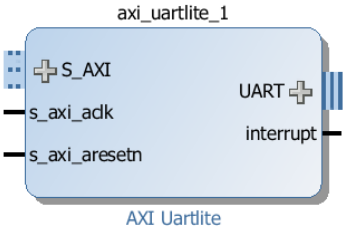
IP jádro Media Independent Interface (MII) to Reduced Media Independent Interface (RMII) poskytuje rozhraní mezi fyzickou vrstvou Ethernetu na desce NEXYS 4 podle standardu IEEE 802.3 a IP jádrem jako je AXI Ethernet Lite, AXI Ethernet nebo Zynq®-7000 PS MAC. Tyto IP jádra poskytují tradiční MII, které vyžaduje 16 signálů pro komunikaci s Ethernetovou fyzickou vrstvou. Jádro MII to RMII přijímá rozhraní 16 signálů MII a poskytuje šesté nebo sedmé signálové rozhraní k RMII. Navíc fixní referenční hodiny s frekvencí 50 MHz synchronizují jádro MII s RMII s oběma rozhraními. Referenční hodiny s frekvencí 50 MHz mohou být poskytovány z krystalu z desky FPGA (po průchodu IP blokem Clock Wizard, jak bylo zmíněno výše) nebo mohou být generovány v hostitelském FPGA. IP jádro MII to RMII je znázorněno na obrázku 26, na kterém je vidět vstup/výstup MII (komunikující s blokem AXI Ethernet Lite), vstupní hodiny a signál sloužící k restartu. Na druhé straně je tedy RMII, které komunikuje s fyzickou vrstvou desky NEXYS 4 [11].



Obrázek 26: Blok MII to RMII

- **AXI UART Lite**

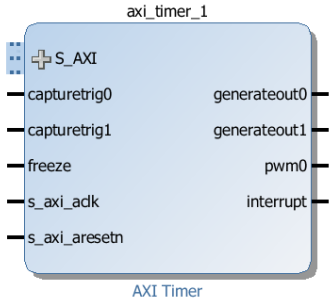
AXI Universal Asynchronous Receiver Transmitter (UART) Lite je IP jádro, které poskytuje rozhraní pro asynchronní přenos sériových dat. Je navrženo tak, aby dokázalo komunikovat podle protokolu AXI4-Lite (toto rozhraní je značeno jako S_AXI na obrázku 27). AXI UART Lite komunikuje ve full duplex režimu, slouží k přenosu dat a přístupu do registrů. Blok má konfigurovatelnou přenosovou rychlost (baud rate), nezávislý vysílač a přijímač FIFO s hloubkou 16 znaků. Blok AXI UART Lite dokáže nezávisle vysílat a přijímat 5 až 8 bitové znaky s jedním stop bitem a paritním bitem (lichým, sudým nebo žádným). Blok serializuje data přijatá skrze rozhraní AXI4 Lite a paralelizuje znaky přijaté z periferií, které jsou sériové. Ve chvíli kdy se přes UART vstup/výstup (na obrázku 27 vpravo) zapisují data do registrů FIFO uvnitř bloku, AXI UART Lite dokáže generovat přerušení. Generování přerušení z tohoto bloku se však v tomto návrhu nevyužívá [12].



Obrázek 27: AXI UART Lite

- **AXI Timer**

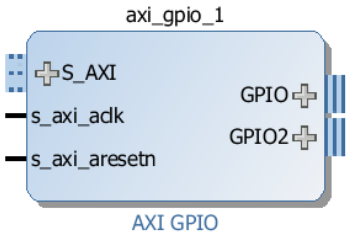
AXI Timer je 32/64 bitový čítač, který je propojen s AXI4-Lite rozhraním. Toto IP jádro má konfigurovatelnou šířku čítače, výstup pulsně šířkové modulace (PWM z angl.. Pulse Width Modulation), dokáže detekovat a generovat události a pozastavit čítače při ladění. V této práci se však z tohoto bloku využívá pouze generování přerušení (interrupt), viz obrázek 28 [13].



Obrázek 28: AXI Timer

- **AXI GPIO**

IP blok AXI GPIO (General Purpose Input/Output) poskytuje základní rozhraní pro komunikaci s periferiemi na desce NEXYS 4 jako jsou LED diody, přepínače, tlačítka atd. Tento blok komunikuje s rozhraním AXI4 Lite a může být nakonfigurován jako jedno nebo dvou kanálové zařízení (GPIO, GPIO2), viz obrázek 29 obsahující 2 kanály. Šířka obou kanálů je nezávisle konfigurovatelná od 1 do 32 bitů. IP jádro AXI GPIO je využito v návrhu dvakrát. První blok slouží pro připojení přepínačů a LED diod a druhý blok slouží pro připojení 7 segmentového displeje [14].



Obrázek 29: AXI GPIO

- **Další IP bloky v logickém obvodu**

Jak už bylo zmíněno, při automatické konfiguraci a propojení některých bloků byly vygenerovány další nezbytné IP bloky. Mezi tyto bloky se řadí lokální paměť procesoru a resetovací obvody jako např. Processor System Reset (rst_clk_wiz_1_100M nebo rst_mig_7_series_0_81M). Dále pak AXI Interrupt Controller, který řídí přerušení, AXI Interconnect, propojující periferie a paměť s procesorem MicroBlaze (microblaze_0_axi_periph a axi_mem_intecon). Všechna tato logická IP jádra jsou využita ve finálním blokovém schématu, uvedeném v příloze 0.

3.2.3 Export do SDK

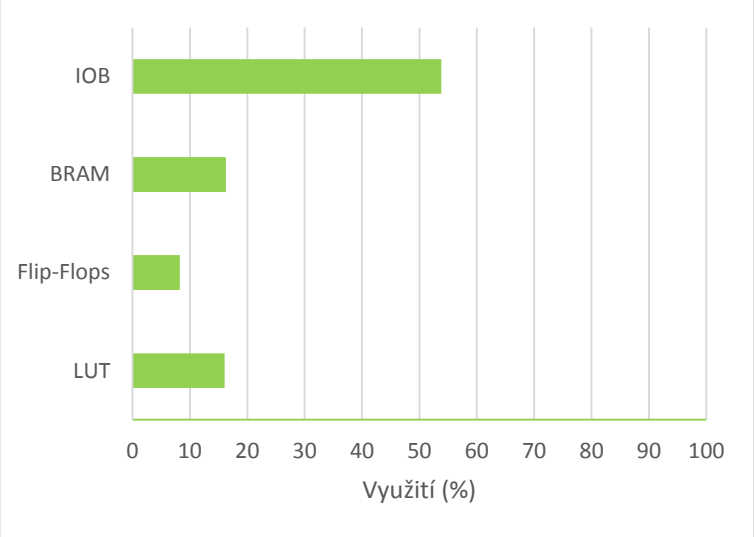
Počet využitých logických prvků z FPGA ve finální verzi logického obvodu ukazuje nadcházející tabulka 6. Mezi tyto logické prvky patří klopné obvody, především typu D (Flip-Flops), vstupně/výstupní bloky (IOB), zajišťující rozhraní mezi signály vnitřní logiky a vnějšími vývody součástky. Dále se pak jedná o tzv. Look-Up Table (LUT), což jsou nástroje pro implementaci

libovolného kombinačního logického obvodu, a o prvky Block RAM (BRAM), které přestavují statickou paměť. Všechny tyto prvky mohou být propojovány pomocí konfigurovatelné propojovací sítě. V následující tabulce je také zahrnuto aktuální procentuální využití prvků se stále dostupným počtem těchto prvků.

Tabulka 6: Využití elementárních logických prvků v obvodu

Prvek	Využité	Dostupné	Využité v %
LUT	10 187	63 400	16,07
Flip-Flops	10 437	126 800	8,23
BRAM	22	135	16,3
IOB	113	210	53,81

Procentuální využití logických prvků v obvodu je také znázorněno následujícím grafem (Obrázek 30).



Obrázek 30: Graf využitých logických prvků

Vytváření aplikačního programu, pro již navržený logický obvod, bylo realizováno ve vývojovém prostředí SDK. Aby však bylo toto vývojové prostředí schopno s logický obvodem pracovat, je nutné ve Vivadu vytvořit tzv. HDL wrapper. Vytvořený HDL wrapper s sebou nese všechny potřebné informace o využitých IP blocích a informace o jejich fyzickém propojení v obvodu FPGA. Na tomto základě je pak SDK schopen stavět aplikační program.

Po vytvoření HDL wrapperu prošel vytvořený HW design syntézou a implementací. Předposledním krokem bylo vygenerování konfiguračního souboru tzv. bitstreamu neboli souboru s příponou bit. Tento bitstream obsahuje binární data, kterými naplní konfigurační paměť.

Posledním krokem v prostředí Vivado byl export HW a bitstreamu. Ve vývojovém prostředí Vivado existuje funkce, která umožňuje přímo spustit prostředí SDK pro právě otevřený projekt. SDK s projektem lze taktéž spustit z vytvořené složky (např. název_projektu.sdk) v hlavním adresáři projektu Vivado.

3.3 Software Development Kit (SDK)

V této kapitole se přesuneme do vývojového prostředí Software Development Kit (dále jen SDK), které slouží pro vývoj softwarových aplikací. SDK je ve své podstatě nezávislý na prostředí Vivado, a od této chvíle může být vyvíjen softwarový projekt v jazyce C nebo C++. Jak bylo zmíněno v předchozí kapitole, tento softwarový projekt je však založen a stavěn na dříve vyexportovaném hardwarovém návrhu (wrapperu), který byl vytvořen ve Vivadu. Tudíž samotný projekt je na prostředí Vivado závislý. Všechny tyto HW specifikace a zahrnuté IP bloky jsou obsaženy v souboru system.hdf.

Pokud by byla nutná jakákoliv modifikace HW specifikací, musel by se pak vytvořit znovu HDL wrapper a soubor bitstream. Následně by tyto soubory musely být znovu exportovány do SDK.

V nadcházejících podkapitolách byla shrnuta realizace aplikačního projektu. Realizace je rozdělena na 3 části, a to na inicializaci, nastavení připojení TCP a demonstraci přenášených dat. Projekt však pracuje s mnoha dalšími zdrojovými soubory, které jsou součástí elektronické přílohy na CD.

3.3.1 Realizace aplikačního projektu - inicializace

Prvním krokem při vytváření aplikačního projektu bylo potřeba si ověřit, zda cílový HW odpovídá našemu vytvořenému HW designu. V tomto případě tedy *design_1_wrapper_hw_platform_0*. Při zakládání projektu byl také vytvořen BSP (board support package), což je balíček, který obsahuje specifické ovladače pro správnou funkčnost používaného HW. Následně byl vybrán vzorový projekt lwIP Echo server, který byl následně modifikován. Po založení projektu se vytvořily dvě další složky. První složka obsahuje zdrojové soubory .c a hlavičkové soubory .h, se kterými budeme následně pracovat. Druhou složkou je již zmíněná složka BSP, která obsahuje soubory pro správnou podporu desky NEXYS4 a implementovaných IP bloků.

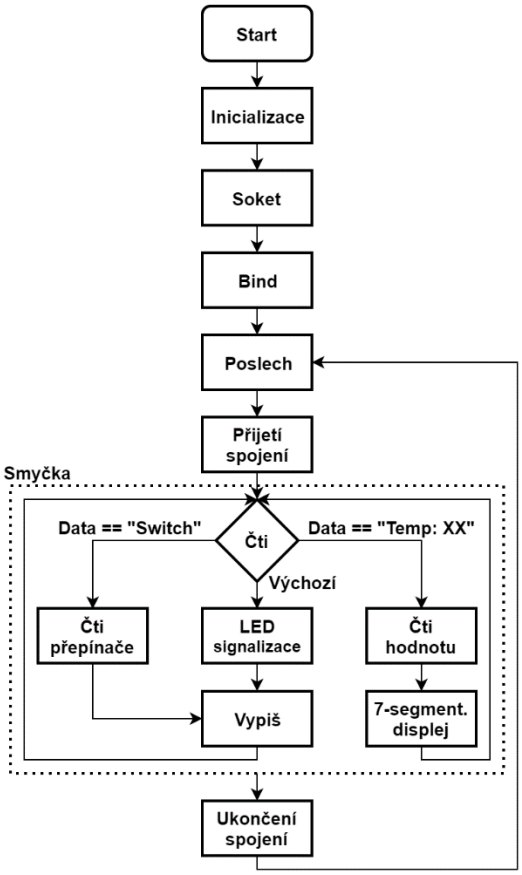
Aplikační projekt zahrnuje vytvoření serveru, který běží na procesoru MicroBlaze. Tento server má za úkol komunikovat s klientem (terminálem v počítači) v intranetové síti. Server zpracovává přijatá data a na jejich základě vykonává určité příkazy. Pro demonstraci správného přenosu dat bylo využito periferií na desce Nexys 4. Konkrétně LED diod, přepínačů a 7 segmentového displeje.

V Ethernetové komunikaci bylo využito kompletního open source lightweight IP (dále jen lwIP) stacku. Stack je využíván především výrobci vestavěných systémů jako je např. Altera, Xillinx, Honeywell atd. LwIP obsahuje protokoly různých vrstev TCP/IP modelu. Základní protokoly se zařazením do jednotlivých vrstev jsou uvedeny v tabulce 7.

Tabulka 7: Protokoly lwIP stacku

Název vrstvy	**Protokoly**
Linková vrstva	ARP, PPP
Síťová vrstva	IP, ICMP, IGMP
Transportní vrstva	TCP, UDP, Raw sockets
Aplikační vrstva	DNS, DHCP, SNMP
Ostatní protokoly	HTTP server, TFTP server, …

Chod aplikačního programu je znázorněn vývojovým diagramem uvedeným na obrázku 31. Po startu programu, ve zdrojovém souboru *main.c*, proběhne celá řada inicializací, mezi které patří inicializace platformy, rychlosti přenosu, lwIP stacku, periferií GPIO, přiřazení MAC adresy, IP adresy, masky podsítě a výchozí brány.



Obrázek 31: Vývojový diagram aplikačního programu (serveru)

Aplikační program echo server umožňuje využití protokolu DHCP (Dynamic Host Configuration Protocol), který je součástí lwIP stacku a používá se pro automatickou konfiguraci jednotlivých stanic připojených do počítačové sítě. V této úloze se však protokol DHCP nepoužívá z důvodu, protože komunikace probíhá v intranetové síti. Server tedy využívá výchozích adres, které je možné změnit. Následuje vytištění hlavičky a přidělených adres do konzole, pomocí příslušných funkcí definovaných ve stejném souboru. Server využívá rychlost přenosu 100 Mbit/s a jeho adresy jsou zobrazeny v obrázku 8. Dále jsou povolena přerušení funkcí *platform_enable_interrupts()* a nastaveno síťové rozhraní příslušnou funkcí.

Tabulka 8: Výchozí adresy serveru

IP adresa	192.168.1.10
Maska podsítě	255.255.255.0
Výchozí brána	192.168.1.1

3.3.2 Realizace aplikačního projektu – nastavení připojení TCP

Po všech úspěšných inicializacích je volána funkce *start_application()*, která je definována v souboru *echo.c*. Nyní se pohybujeme v obrázku 31 od bloku *Soket,* až po blok *Přijetí spojení*. Funkce *start_application()* vrací hodnotu datového typu integer, která představuje chybové hlášky (definice errorů lze najít v souboru *err.h*). Zde se pracuje se strukturou neboli řídícím blokem PCB (Protocol Control Block) a volají se zde funkce jako *tcp_new*, *tcp_bind, tcp_listen* a *tcp_accept*. Tyto funkce zde byly stručně popsány.

Prvně byla deklarována struktura *tcp_pcb*, do které je zapisováno ukazatelem *pcb.* Tato struktura představuje koncový bod komunikace a je deklarována v hlavičkovém souboru *tcp.h*. Jedná se zde tedy o vytvoření nového řídícího bloku připojení (soketu), který obsahuje informace o protokolech, pomocí kterých bude přenos dat zajištěn. Vytvoření bloku PCB je realizováno pomocí funkce *tcp_new()*, viz nadcházející zdrojový kód v obrázku 32. Není-li dostatek paměti pro vytvoření PCB je vypsána chybová hláška a vrácena hodnota -1 označující nedostatek paměti (hodnota je k nalezení v již zmíněném souboru *err.h*).

```
pcb = tcp_new();
if (!pcb) {
	xil_printf("Error creating PCB. Out of Memory\n\r");
	return -1;
}
```

Obrázek 32: Založení řídícího bloku PCB

Po tomto kroku vznikne spojení soketu s lokálním portem, jedná se o tzv. nabindování pomocí funkce *tcp_bind*, do které vstupuje právě vytvořená struktura *tcp_pcb* (respektive jen její ukazatel *pcb*), IP adresa a port, viz obrázek 33. Pokud návratovou hodnotou funkce *tcp_bind* není *0* (*0 = ERR_OK*), je vypsána chybová hláška. Jedná se např. o případ snahy propojit soket s již používaným portem (*ERR_USE*).

```
err = tcp_bind(pcb, IP_ADDR_ANY, port);
if (err != ERR_OK) {
	xil_printf("Unable to bind to port %d: err = %d\n\r", port, err);
	return -2;
}
```

Obrázek 33: Propojení PCB s lokálním portem

Konkrétně se jedná o 7 port, na kterém bude server poslouchat pro přicházející spojení pomocí funkce *tcp_listen()*, viz *poslech* na obrázku 31. Tato funkce vrací nový PCB s tím, že vstupující blok PCB do této funkce je dealokován. Tento krok slouží k ušetření paměti. Za touto funkcí musí být volána funkce *tcp_accept()*, zobrazena v obrázku 34. Pokud tato funkce zavolána nebude, pak všechna přicházející spojení budou přerušena. Ve chvíli, kdy se na server připojí klient, následuje přijetí spojení *(*již zmíněná funkce *tcp_accept)*. Pokud během těchto funkcí nenastala žádná chyba, funkce *start_application* vrací hodnotu *0* a program se dostává do smyčky.

```
pcb = tcp_listen(pcb);
if (!pcb) {
	xil_printf("Out of memory while tcp_listen\n\r");
	return -3;
}

tcp_accept(pcb, accept_callback);
```

Obrázek 34: Funkce pro poslech a přijetí spojení

3.3.3 Realizace aplikačního projektu – demonstrace přenesených dat

V tomto okamžiku se program dostává do nekonečné smyčky (v souboru *main.c*), ze které se dostane ven jedině v případě, nastane-li ukončení spojení s klientem *close* v obrázku *31*. Pak se aplikační program dostává opět do fáze poslech, kdy poslouchá pro přicházející spojení.

Jakmile je navázáno spojení s klientem, je umožněn přenos dat mezi jednotlivými stanicemi (*smyčka* v obrázku 31). V této smyčce je nutné v pravidelných intervalech volat funkci *xemacif_input()*, jejíž úkolem je vytáhnout pakety z RX Bufferu IP bloku AXI Ethernet Lite (viz podkapitola 3.2.2 – AXI Ethernet Lite) a předat je stacku lwIP pro další zpracování. Klient nyní může odeslat data serveru, na základě kterých pak server vykoná určitý příkaz. Objeví-li se aktivita na fyzické vrstvě (odesláním dat klientem), vybudí se přerušení a aplikační program zpracuje příchozí paket podle stanovených protokolů a přečte si informaci. Zpracování přijatých dat a následné vykonání určitého příkazu se vykonává ve funkci *recv_callback* ve zdrojovém souboru *echo.c* v příloze na CD.

Příchozí data jsou uložena do instance (člena) *payload* struktury *pbuf*, která mimo jiné obsahuje také informace o pořadí příchozích dat, jejich velikosti atd. Celá tato struktura má svou deklaraci v hlavičkovém souboru *pbuf.h*. Přijatá data uložená v *payload* jsou nyní porovnávána za pomoci podmínky a funkce *strncmp*, která porovnává řetězce. Pokud se porovnávaný řetězec shoduje s příchozí zprávou, pak je hodnota této funkce rovná nule.

Je-li příchozí informace „Switch“, viz obrázek 35, pak program přečte aktuální hodnotu přepínačů na desce Nexys 4, uloží ji do námi vytvořeného pole (*buffer*) a následně tuto hodnotu odešle klientovi v dekadické podobě pomocí funkce *tcp_write*. Celá odpověď serveru má tvar: „Dekadická hodnota switchů je: X“, kde X je dekadická hodnota přepínačů nabývající hodnot od 0 do 65535. Maximální hodnota je dána počtem všech sepnutých spínačů na Nexys 4 ($2^{16} = 65535$).

```
if(strncmp(p->payload,"Switch",6) == 0){
		sprintf(buffer, "%d", sw);
		err = tcp_write(tpcb, "\nDekadicka hodnota switchu je: ", 32, 1);
		err = tcp_write(tpcb, buffer, strlen(buffer), 1);
		err = tcp_write(tpcb, "\n", 2, 1);
}
```

Obrázek 35: Porovnávání řetězce a vykonání příkazu

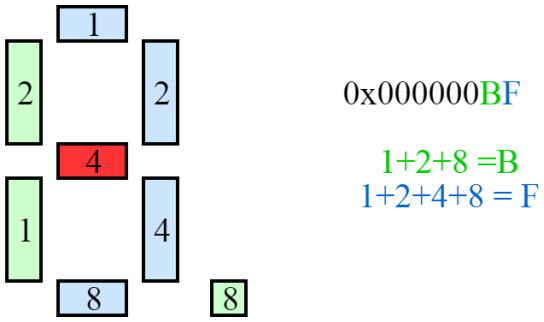
V případě, že serveru dorazí klíčové slovo „Temp YY“, kde YY je dvouciferná hodnota teploty. Server tuto informaci zpracuje a přijatou hodnotu zobrazí na 7 segmentovém displeji ve tvaru např.: „23°C“. Přijatá data jsou porovnána stejným způsobem, jako v předcházejícím případě. Aby bylo možné vykreslit informaci dvouciferné hodnoty na 7 segmentový displej, je nutné si vytvořit dvě proměnné, do kterých budou zapsány zvlášť desítky a zvlášť jednotky (*dessw* a *jedsw*). Tyto znaky jsou následně porovnány pomocí funkce *switch*, a na základě shodnosti s danými případy je globálním proměnným *des* a *jed* přidělena hodnota, která slouží pro zobrazení na displeji, viz obrázek 36. Stane-li se, že

za klíčovým slovem nebude následovat dvouciferná hodnota, ale jakékoli jiné znaky, bude vybrána hodnota (představující pomlčku) pod případem *default*.

```
switch(jedsw){
case '0':
      jed = 0x000000C0; //hodnota 0 na displeji
      break;
case '1':
      jed = 0x000000F9; //hodnota 1 na displeji
      break;
case '2':
      jed = 0x000000A4; //hodnota 2 na displeji
      break;
case '3':
      jed = 0x000000B0; //hodnota 3 na displeji
      break;
case '4':
      jed = 0x00000099; //hodnota 4 na displeji
      break;
case '5':
      jed = 0x00000092; //hodnota 5 na displeji
      break;
case '6':
      jed = 0x00000082; //hodnota 6 na displeji
      break;
case '7':
      jed = 0x000000F8; //hodnota 7 na displeji
      break;
case '8':
      jed = 0x00000080; //hodnota 8 na displeji
      break;
case '9':
      jed = 0x00000090; //hodnota 9 na displeji
      break;
default :
      jed = 0x000000BF; //hodnota pomlcky na displeji
}
```

Obrázek 36: Přiřazení hodnoty globální proměnné „jed“ v závislosti na příchozích datech

Pro přiřazení hodnoty globální proměnné *des* je předcházející kód obdobný. Za touto funkcí následuje podmínka, která vyhodnotí, zda oba znaky za klíčovým slovem jsou číslice. Pokud ano, přiřadí globálním proměnným *stu* a *cel* příslušné hodnoty tak, aby se na displeji zobrazilo *„°C“*. Pokud však alespoň jeden znak není číslice, bude všem globálním proměnným (*des, jed, stu* a *cel*) přiřazena hodnota *„0x000000BF“* představující pomlčku na displeji, viz obrázek 37.



Obrázek 37: Princip přiřazování hodnot pro 7 segmentový displej

Samotné zobrazování bylo realizováno ve zdrojovém souboru *main.c* v nekonečné smyčce, za pomoci cyklů *for*. Důvodem využití těchto cyklů je ten, že 7 segmentový displej má jednu společnou anodu, tudíž musí být v jednom okamžiku vykreslován pouze jeden znak. V cyklech byla také využita proměnná *a* datového typu integer, která je postupně inkrementována. Pokud je tedy tato proměnná menší než

hodnota 1000 rozsvěcuje jeden znak displeje po dobu 1 ms (přesněji pozici jednotek). Překročí-li hodnota proměnné *a* hodnotu 1000, pak bude 1 ms vykreslována pozice desítek atd. (Obrázek 38).

```
for(a = 0; a < 1000; a++){
XGpio_DiscreteWrite(&gpSEG, 1, jed);          //zapis aktualniho znaku
XGpio_DiscreteWrite(&gpSEG, 2, 0x000000FB); //anoda - pozice - jednotky
}
for(a = 1000; a < 2000; a++){
XGpio_DiscreteWrite(&gpSEG, 1, des);
XGpio_DiscreteWrite(&gpSEG, 2, 0x000000F7); //anoda - pozice - desitky
}
for(a = 2000; a < 3000; a++){
XGpio_DiscreteWrite(&gpSEG, 1, stu);
XGpio_DiscreteWrite(&gpSEG, 2, 0x000000FD); //anoda - pozice - stupne
}
for(a = 3000; a < 4000; a++){
XGpio_DiscreteWrite(&gpSEG, 1, cel);
XGpio_DiscreteWrite(&gpSEG, 2, 0x000000FE); //anoda - pozice - celsia
}
a = 0;
```

Obrázek 38: Cykly pro vykreslování znaků na 7 segmentovém displeji

Z obrázku je patrné, že do cyklů vstupují globální proměnné ze souboru *echo.c*. Aby však byly tyto hodnoty zde viditelné, bylo potřeba založit hlavičkový soubor *echo.h*, kde byly proměnné deklarovány, a také bylo potřeba do souboru *main.c* tento hlavičkový soubor zahrnou pomocí *#include*.

Pokud však klient odešle jiná data, než tato klíčová slova, server pouze signalizuje přijetí dat pomocí vestavěných LED diod na Nexys 4, které se na vteřinu rozsvítí, a poté se pošle odesílateli informace o tom, aby zadal klíčové slovo.

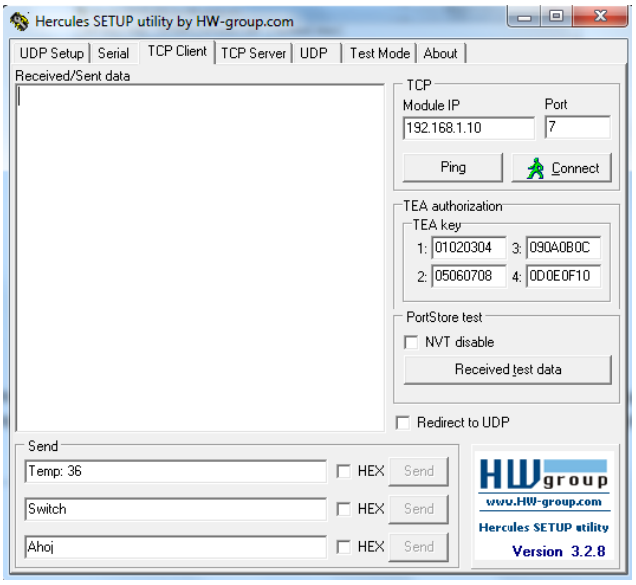
4 Testování demonstrační úlohy

V této kapitole je popsáno testování přenosu dat mezi počítačem a deskou FPGA Nexys 4. Testovala se především správná funkce logického obvodu a vytvořeného serveru, který určitým způsobem reaguje na příchozí informace, viz kapitola 3.3.1. Dále proběhlo testování doby odezvy serveru (ping), které bylo statisticky zhodnoceno.

4.1 Ověření správnosti přenesených dat

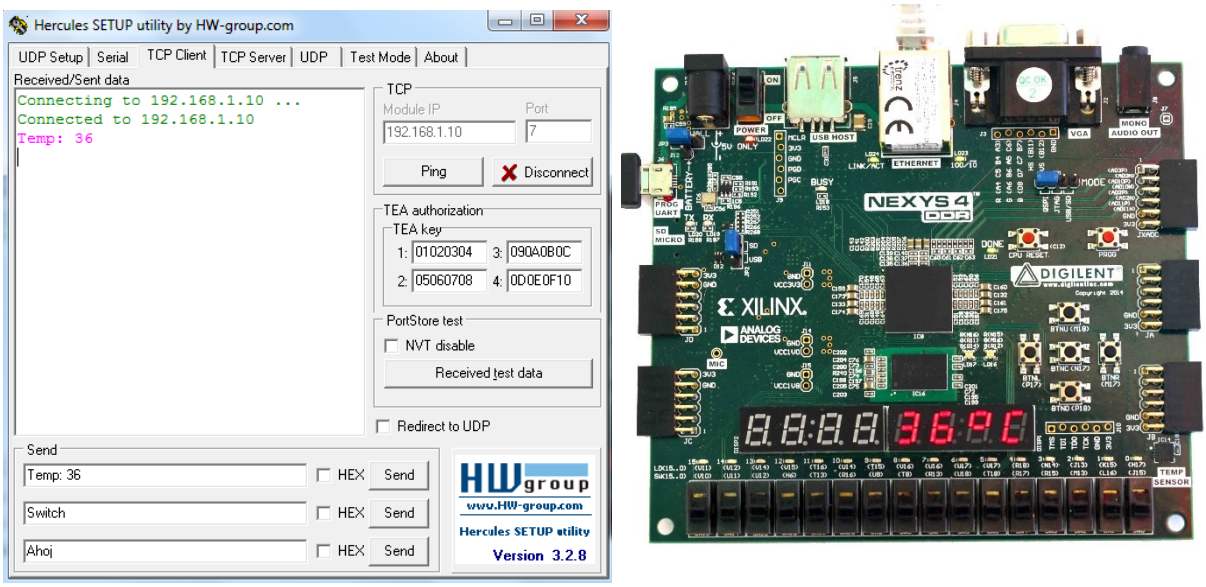
Komunikace se serverem byla testována pomocí dvou terminálů. Konkrétně, volně dostupným terminálem Tera Term a terminálem Herkules SETUP vytvořeným společností HW group. Práce s oběma programy funguje na stejném principu, proto zde bude popsáno pouze testování s terminálem Herkules.

Dříve než byl spuštěn vytvořený aplikační program (server), bylo nutné nakonfigurovat FPGA. Konfiguraci logického obvodu je možné provést přímo ve vývojovém prostředí SDK za pomoci funkce *Program FPGA*. V této funkci byl vybrán již vytvořený HW návrh design_1_wrapper_hw_platform_0 a vygenerovaný bitstream, viz kapitola 3.2.3. Po nakonfigurování je možné spustit aplikační program, jehož chod byl popsán v kapitole 3.3.1. Vytvořený server nyní poslouchá na 7 portu a jeho IP adresa je 192.168.1.10. V tuto chvíli může být se serverem navázáno spojení. Spojení realizujeme pomocí zmíněného terminálu Herkules, který zastupuje roli TCP klienta. Z názvu je zřejmé využití TCP protokolu, jehož funkce byla popsána v 1.4.3. Na následujícím obrázku je zobrazeno GUI (grafické uživatelské rozhraní) terminálu (Obrázek 39). Na levé straně je okno, sloužící pro zobrazení odeslaných a přijatých dat. Pod tímto oknem jsou tři textboxy, ve kterých jsou připraveny tři varianty dat, které budeme serveru posílat. Na pravé straně terminálu byla zadána IP adresa společně s číslem portu a pomocí tlačítka „Connect“, bylo navázáno spojení.



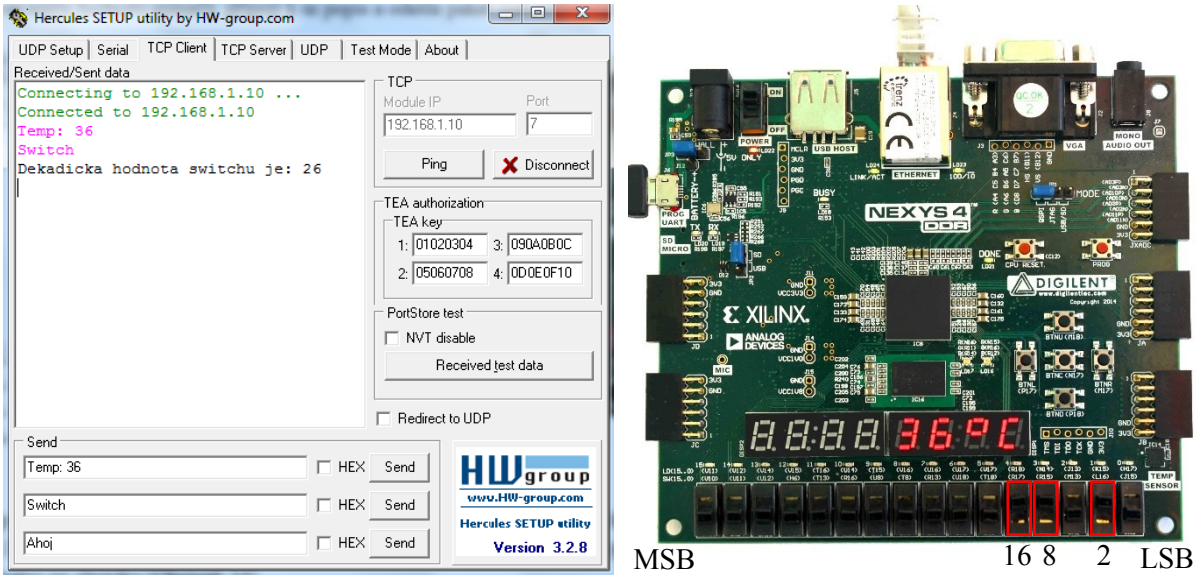
Obrázek 39: GUI Terminálu Herkules SETUP

V této chvíli je možné posílat data serveru. Prvním klíčovým slovem, na které server reaguje je: „Temp: 36“. Server v tuto chvíli vypíše hodnotu za klíčovým slovem na sedmi segmentový displej, viz obrázek 40. Princip odesílání dat je shodný s dříve uvedeným obrázkem 16.



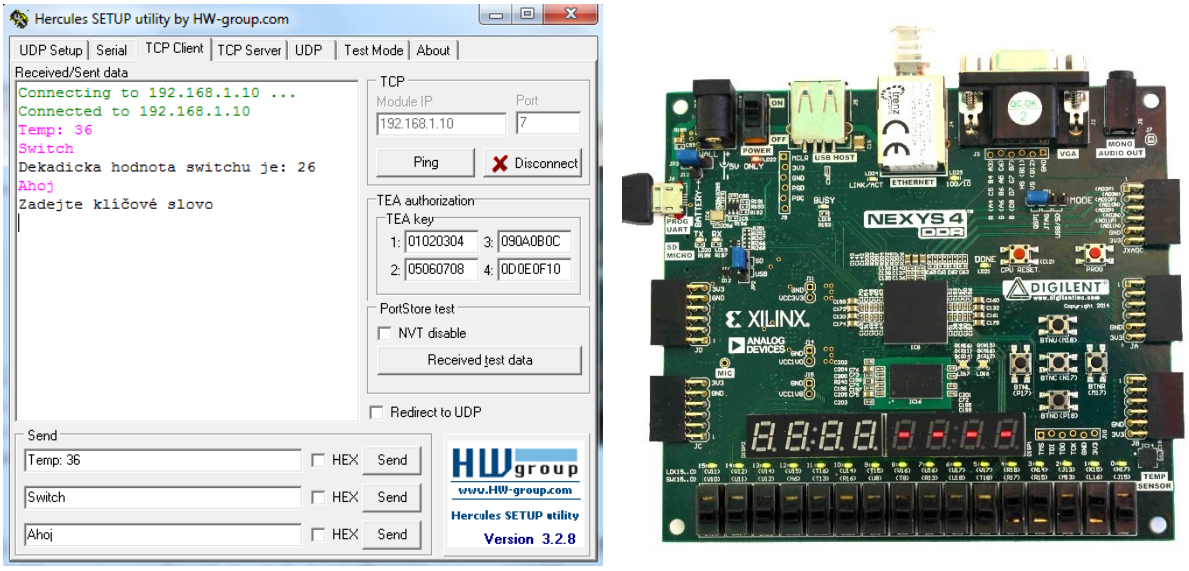
Obrázek 40: Demonstrační úloha – 1

Nyní byly na desce Nexys 4 sepnuty některé ze spínačů, viz obrázek 41 vpravo dole. První přepínač zprava, je spínač s nejméně významným bitem (LSB – může nabývat hodnoty 0 nebo 1), a poslední zprava je spínač s nejvýznamnějším bitem (MSB – může nabývat hodnoty 0 nebo 32768). Z obrázku je patrná hodnota spínačů 26 (16 + 8 + 2). Po odeslání klíčového slova „Switch“ z terminálu (značeno fialově), server načte hodnotu spínačů, připojí k ní popis a odešle paket zpět klientovi.



Obrázek 41: Demonstrační úloha – 2

Pokud však přijdou serveru jiná data, než již zmíněná klíčová slova, pak server na přijatá data reaguje rozsvícením LED diod, vymazáním sedmi segmentového displeje a odesláním zprávy odesílateli, aby zadal klíčové slovo, jak je znázorněno na obrázku 42.



Obrázek 42: Demonstrační úloha – 3

Při testování demonstrační úlohy byla také otestována správná funkce protokolu TCP. Tento protokol zaručuje spolehlivost přenosu dat, viz kapitola 1.4.3. Po navázání spojení klienta se serverem byl odpojen Ethernetový kabel a následně odeslána data. V tuto chvíli odesílatel čekal na kladné potvrzení o přijetí paketu druhou stranou, které nemohl obdržet, a tudíž po náhodném časovém intervalu se snažil odeslat data znovu. Po opětovném připojení přenosového média byl paket úspěšně přenesen. V případě využití protokolu UDP, by byl paket nenávratně zahozen.

4.2 Testování funkčnosti spojení

Ověření funkčnosti spojení mezi počítačem a vytvořeným serverem na přípravku Nexys 4 bylo také provedeno za pomoci měření doby odezvy. Program ping (z angl. Packet InterNet Groper) měří tzv. dobu obrátky (RTT – round trip time), což je doba, za kterou paket urazí cestu k příjemci a zpět k odesílateli.

Měření bylo provedeno za pomoci příkazového řádku (Command Prompt), který je součástí OS Windows. Zde byl zadán příkaz ping s IP adresou serveru, tzn. „ping 192.168.1.10 –t “. Každý odesílaný paket měl velikost 32 Bytů. Pro zjištění průměrné doby odezvy bylo provedeno sto měření, přičemž žádný z odeslaných paketů nebyl během přenosu ztracen. Průměrná doba odezvy tedy byla 2 ms. Maximální a minimální doba odezvy je zaznačená v tabulce 9. Celý výpis jednotlivých měření je uveden v elektronické příloze na CD.

Tabulka 9: Měření doby odezvy

Odeslané pakety	100
Přijaté pakety	100
Ztracené pakety	0
Minimální doba odezvy	< 1 ms
Maximální doba odezvy	10 ms
Průměrná doba odezvy	2 ms

Závěr

V této bakalářské práci byl navržen a implementován logický obvod pro přenos dat po síti Ethernet na vývojové desce FPGA Digilent Nexys 4 DDR s přenosovou rychlostí 100 Mbit/s. Logický obvod byl realizován pomocí IP bloků, s využitím vestavěného procesorového jádra MicroBlaze. Na tomto implementovaném procesoru byl realizován aplikační program v jazyce C – server, na jehož IP adresu a port se připojil vzdálený počítač. Komunikace po této intranetové síti využívá komunikačních protokolů z rodiny protokolů TCP/IP, které jsou součástí kompletního open-source lwIP stacku, navrženého právě pro vestavěné systémy.

Cílem demonstrační úlohy bylo reprezentovat správnost přenesených dat. Pro splnění tohoto cíle bylo využito periferií přípravku Nexys 4, kde server na základě příchozích informací od klienta, z těchto periferií čte nebo do nich zapisuje. Konkrétně se jedná o odeslání hodnoty sepnutých spínačů vzdálenému počítači a o výpis dvouciferného čísla na 7 segmentový displej v závislosti na přijeté informaci. Při testování demonstrační úlohy byl využit terminál Herkules jako TCP klient, pomocí kterého bylo vytvořeno spojení se serverem a kontrolována správnost přenesených dat. V poslední řadě byla otestována doba odezvy serveru pomocí funkce ping, která byla v průměru 2 ms.

Úloha zprostředkovává obousměrnou komunikaci na úrovni transportní vrstvy mezi počítačem a FPGA za pomoci protokolů TCP a IP. Díky použitému protokolu TCP je zajištěna spolehlivost přenosu tzv. kladným potvrzováním, kdy příjemce posílá odesílateli zprávu o tom, že data přijal. Zadání bakalářské práce bylo úspěšně splněno a výsledná demonstrační úloha je plně funkční. Celá úloha, jak logický obvod, tak i aplikační program, je k dispozici v elektronické příloze na CD.

Zdroje

[1] DOSTÁLEK, Libor a Alena KABELOVÁ. *Velký průvodce protokoly TCP/IP a systémem DNS*[online]. 2. Praha: Computer Press, 2000. ISBN 80-7226-323-4. Dostupné z: http://download.matus.in/it/Velky%20pruvodce%20protokoly%20TCP-IP%20a%20systemem%20DNS.pdf

[2] IEEE 802.3-2015. *IEEE Standard for Ethernet*. Revision of IEEE Std 802.3-2012. New York, NY: IEEE Standards Association, 2015

[3] *Nexys4™ FPGA Board Reference Manual* [online]. Digilentinc, 2013, str. 29. Dostupné z: https://www.xilinx.com/support/documentation/university/XUP%20Boards/XUPNexys4/documenatation/Nexys4_RM_VB1_Final_3.pdf

[4] Rodina protokolů TCP/IP verze 3: Protokol IPv4. *SlideShare* [online]. Praha: Jiří Peterka, 2015, 2014. Dostupné z: http://www.earchiv.cz/a92/a231c110.php3

[5] Síťový model TCP/IP. *EArchiv.cz* [online]. Jiří Peterka, 2015, 2015. Dostupné z: http://www.earchiv.cz/a92/a231c110.php3

[6] ZEZULKA, František prof. Ing. CSc. *Průmyslový Ethernet II:: Referenční model ISO/OSI* [online]. 2007, str.86-90. Dostupné z: http://www.uamt.feec.vutbr.cz/~zezulka/download/KPPA/A030786_II.pdf

[7] O'Reilly: Full-Duplex Ethernet. *O'Reilly: Full-Duplex Ethernet* [online]. Safari Books Online, 2017, 2017. Dostupné z: https://www.safaribooksonline.com/library/view/ethernet-the-definitive/1565926609/ch04.html

[8] DIGILENT, Inc. *Pmod NIC100: Network Interface Controller* [online] Dostupné z: http://store.digilentinc.com/pmod-nic100-network-interface-controller/

[9] DIGILENT, Inc. *Nexys 4 DDR - Getting Started with Microblaze Servers.* [online] Dostupné z: https://reference.digilentinc.com/learn/programmable-logic/tutorials/nexys-4-ddr-getting-started-with-microblaze-servers/start

[10] Xilinx, Inc. *AXI Ethernet Lite MAC v3.0: LogiCORE IP Product Guide* [online]. 18.11.2015. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_ethernetlite/v3_0/pg135-axi-ethernetlite.pdf

[11] Xilinx, Inc. *MII to RMII v2.0: LogiCORE IP Product Guide* [online]. 5.4.2017. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/mii_to_rmii/v2_0/pg146-mii-to-rmii.pdf

[12] Xilinx, Inc. *AXI UART Lite v2.0: LogiCORE IP Product Guide* [online]. 5.4.2017. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_uartlite/v2_0/pg142-axi-uartlite.pdf

[13] Xilinx, Inc. *AXI Timer v2.0: LogiCORE IP Product Guide* [online]. 5.8.2016. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_timer/v2_0/pg079-axi-timer.pdf

[14] Xilinx, Inc. *AXI GPIO v2.0: LogiCORE IP Product Guide* [online]. 5.8.2016. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_gpio/v2_0/pg144-axi-gpio.pdf

[15] XILINX. *LwIP 1.4.1 Library (v1.3)* [online]. In: . 18.11.2015, s. 16. Dostupné z: https://forums.xilinx.com/xlnx/attachments/lwip141_v1_7.pdf

[16] *LwIP 2.0.2: Lightweight IP stack* [online]. 1.8.13n. l. Dostupné z: http://www.nongnu.org/lwip/2_0_x/index.html

[17] KAŠÍK, Vladimír. *Programování hradlových polí.* Učební text a návody do cvičení. VŠB-TU Ostrava, 2012.

[18] Porovnání protokolů IPv4 a IPv6. *IBM: IBM Knowledge Center* [online]. Praha, 2017. Dostupné z: https://www.ibm.com/support/knowledgecenter/cs/ssw_ibm_i_72/rzai2/rzai2compipv4ipv6.htm

[19] BC. GREGAR, Pavel. *Ethernetové rozhraní na prípravku Nexys4 v jazyce VHDL* [online]. Praha, 2017. Dostupné z: https://dspace.cvut.cz/handle/10467/68592. Diplomová práce. České vysoké učení technické v Praze. Vedoucí práce Ing. Pavel Lafata Ph.D.

[20] OSI model. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017, 2017. Dostupné z: https://en.wikipedia.org/wiki/OSI_model

[21] KUBÍČEK, Miroslav. *Přenos dat po Ethernetu mezi vývojovými deskami FPGA* [online]. Praha, 2015. Dostupné z: https://dspace.cvut.cz/handle/10467/61128. Bakalářská práce. České vysoké učení technické v Praze. Vedoucí práce Ing. Richard Šusta, Ph.D.

[22] Ethernet. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017. Dostupné z: https://cs.wikipedia.org/wiki/Ethernet

[23] *Medical Imaging Implementation Using FPGAs* [online]. 2010, str. 10. Dostupné z: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-medical.pdf

[24] *Brain activity monitoring system with HDMI interface developed on an FPGA* [online]. Pakistan: IEEE, 2017, 2017. DOI: 10.1109/ICET.2016.7813272. Dostupné z: http://ieeexplore.ieee.org/document/7813272/

[25] *Highly scalable and flexible FPGA based platform for advanced ultrasound research* [online]. Germany: IEEE, 2013, 2013, str. 6. DOI: 10.1109/ULTSYM.2012.0519. Dostupné z: http://ieeexplore.ieee.org/document/6562265/

[26] *Brain activity monitoring system with HDMI interface developed on an FPGA* [online]. India: IEEE, 2015, str. 5. DOI: 10.1109/ICSPCom.2015.7150618. Dostupné z: http://ieeexplore.ieee.org/document/7150618/

[27] Webopedia: CSMA/CD. *Webopedia: CSMA/CD* [online]. 2017. Dostupné z: http://www.webopedia.com/TERM/C/CSMA_CD.html

[28] Telemedicína. *WikiSkripta* [online]. Praha, 2016]. Dostupné z: http://www.wikiskripta.eu/index.php/Telemedic%C3%ADna

Seznam příloh

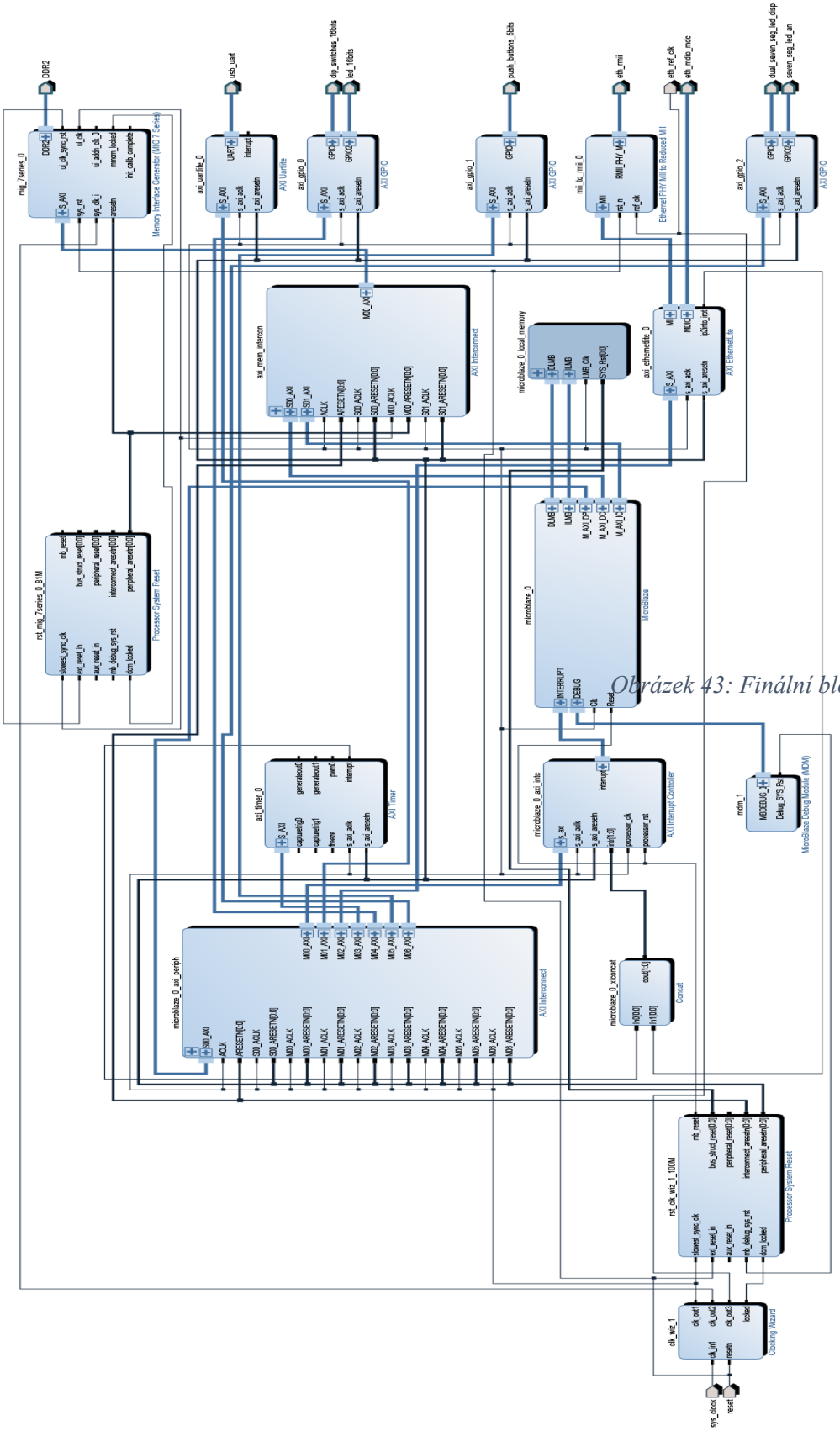
Příloha I: Finální blokové schéma logického obvodu

Příloha II: Příloha na CD, viz tabulka 10

Tabulka 10: Obsah přiloženého CD

Název adresáře	**Popis**
Bakalarska_prace	Bakalářská práce v elektronické podobě
Schema	Finální blokové schéma logického obvodu
Demonstracni_uloha	Kompletní projekt s návodem na spuštění
Testovani_odezvy	Jednotlivá měření doby odezvy serveru

Příloha I



Obrázek 43: Finální blokové schéma logickéh